PCT/GB04/02568

INVESTOR IN PEOPLE

The Patent Office
Concept House
Cardiff Road
Newport
South Wales
NP10 8QQ

I, the undersigned, being an officer duly authorised in accordance with Section 74(1) and (4) of the Deregulation & Contracting Out Act 1994, to sign and issue certificates on behalf of the Comptroller-General, hereby certify that annexed hereto is a true copy of the documents as originally filed in connection with the patent application identified therein.

In accordance with the Patents (Companies Re-registration) Rules 1982, if a company named in this certificate and any accompanying documents has re-registered under the Companies Act 1980 with the same name as that with which it was registered immediately before re-registration save for the substitution as, or inclusion as, the last part of the name of the words "public limited company" or their equivalents in Welsh, references to the name of the company in this certificate and any accompanying documents shall be treated as references to the name with which it is so re-registered.

In accordance with the rules, the words "public limited company" may be replaced by p.l.c., plc, P.L.C. or PLC.

Re-registration under the Companies Act does not constitute a new legal entity but merely subjects the company to certain additional company law rules.

REC'D 02 AUG 2004
WIPO PCT

Signed [signature]

Dated 21 July 2004

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN COMPLIANCE WITH RULE 17.1(a) OR (b)

Patents Form 1/77

Patents Act 1977
(Rule 16)

1/77

Request for grant of a patent

(See the notes on the back of this form. You can also get an explanatory leaflet from the Patent Office to help you fill in this form)

THE PATENT OFFICE
JR
13 JUN 2003
RECEIVED BY FAX

The Patent Office
Cardiff Road
Newport
South Wales
NP10 8QQ

1. Your reference

AN IMPROVED TELEVISION SYSTEM

2. Patent application number
(The Patent Office will fill in this part)

13 JUN 2003

0313720.5

3. Full name, address and postcode of the or of each applicant *(underline all surnames)*

ELECTRA GUIDE LIMITED (ENGLAND)

Patents ADP number *(if you know it)* 08659255400

If the applicant is a corporate body, give the country/state of its incorporation

4. Title of the invention

AN IMPROVED TELEVISION SYSTEM

5. Name of your agent *(if you have one)*

"Address for service" in the United Kingdom to which all correspondence should be sent *(including the postcode)*

Patents ADP number *(if you know it)*

6. If you are declaring priority from one or more earlier patent applications, give the country and the date of filing of the or of each of these earlier applications and *(if you know it)* the or each application number

Country	Priority application number *(if you know it)*	Date of filing *(day / month / year)*

7. If this application is divided or otherwise derived from an earlier UK application, give the number and the filing date of the earlier application

Number of earlier application	Date of filing *(day / month / year)*

8. Is a statement of inventorship and of right to grant of a patent required in support of this request? *(Answer 'Yes' if:*

a) any applicant named in part 3 is not an inventor, or
b) there is an inventor who is not named as an applicant, or
c) any named applicant is a corporate body.
See note (d))

YES

Patents Form 1/77

following items you are filing with this form.
Do not count copies of the same document

Continuation sheets of this form	
Description	26 pages
Claim(s)	not filed
Abstract	not filed
Drawing(s)	37 figures

10. If you are also filing any of the following, state how many against each item.

Priority documents	not filed
Translations of priority documents	not filed
Statement of inventorship and right to grant of a patent *(Patents Form 7/77)*	included
Request for preliminary examination and search *(Patents Form 9/77)*	not filed
Request for substantive examination *(Patents Form 10/77)*	not filed
Any other documents *(please specify)*	not filed

11. I/We request the grant of a patent on the basis of this application.

Signature [signature] Date 13 June 2003

12. Name and daytime telephone number of person to contact in the United Kingdom: Jonathan DRAZIN 07767 688158

Warning

After an application for a patent has been filed, the Comptroller of the Patent Office will consider whether publication or communication of the invention should be prohibited or restricted under Section 22 of the Patents Act 1977. You will be informed if it is necessary to prohibit or restrict your invention in this way. Furthermore, if you live in the United Kingdom, Section 23 of the Patents Act 1977 stops you from applying for a patent abroad without first getting written permission from the Patent Office unless an application has been filed at least 6 weeks beforehand in the United Kingdom for a patent for the same invention and either no direction prohibiting publication or communication has been given, or any such direction has been revoked.

Notes

a) *If you need help to fill in this form or you have any questions, please contact the Patent Office on 08459 500505.*

b) *Write your answers in capital letters using black ink or you may type them.*

c) *If there is not enough space for all the relevant details on any part of this form, please continue on a separate sheet of paper and write "see continuation sheet" in the relevant part(s). Any continuation sheet should be attached to this form.*

d) *If you have answered 'Yes' Patents Form 7/77 will need to be filed.*

e) *Once you have filled in the form you must remember to sign and date it.*

f) *For details of the fee and ways to pay please contact the Patent Office.*

An Improved Interactive Television System

The present invention relates to an interactive television system that downloads a conditional hierarchy of data objects into electronic memory within a television, set-top-box or some other consumer electronic appliance that receives and displays a television service.

Current situation and limitations of present systems

Interactive services to digital television and set-top-box platforms have been in use for a number of years. These generally allocate volatile memory, usually dynamic random access memory (DRAM), within the platform to temporary storage of an interactive software application. Users may invoke the application by responding to a cue displayed on the platform's screen, causing the platform to download the application to the storage area from a continuously broadcast "carousel" of data. Normally the application is split into multiple objects, some comprising program executables, with others comprising data objects including graphics, video, text and sound objects. Each object may link to other objects. Normally, the application persists in storage until the user selects another application cue, whereupon it is either wholly or partially overwritten by the new application so, that if the user selects it again, the application must be again downloaded. Often, the application's objects are broadcast alongside a television service so that they share the same multiplex. In which case, the cue is often displayed over the television video in a format or style specific to the application.

There are a number of problems associated with this approach. First, the user experiences an undesired delay between selecting the cue and the application starting up while the application's objects are downloaded into the platform. Second, the broadcaster must repetitively broadcast the same application, usually many hundreds or thousands of times during the day. This is wasteful of bandwidth, and is particularly costly to terrestrial broadcasters whose available bandwidth is often limited compared to their cable and satellite counterparts.

An alternative approach is to store the application in the platform in non-volatile memory (often referred to as "flash memory") as "firmware" during manufacture.

This has the obvious advantage that the application is pre-stored within the platform, and hence more responsive, but the obvious disadvantage that many types of interactive application become out of date almost immediately or during the platform's service life. An alternative is to download to the platform's flash memory new versions of the application, or different applications, throughout the platform's service life. There are drawbacks with this approach also. First, non-volatile flash memory is more expensive, size for size, when compared to volatile DRAM – making flash firmware implementations generally more expensive compared to DRAM based memory. Second, data written to and recovered from storage using the cheaper (so called "NAND") forms of flash memory are occasionally corrupted and unreliable, and must be read serially into DRAM for error detection/correction first before they can be executed or processed – with a result that large amounts of DRAM are required in any case to host the application's code and data during execution.

Another trend in provision of interactive television services is to broadcast only a single interactive application object which is downloaded into all types of platform, irrespective of their storage capacity, brand or model identity, or other platform parameters such as type of central processor unit (CPU) or platform location. Most such applications are broadcast in so called "middleware" formats that are independent of a platform's hardware or CPU type, such as MHEG-5, JavaTV, MHP (Multimedia Home Platform), MediaHighway or OpenTV, and interpreted locally. Generally, there is limited interoperability between these middleware formats so that, for example, a platform configured or "ported" to play OpenTV applications cannot simultaneously play, say, a MediaHighway application. The growing number of incompatible middleware formats has introduced uncertainties and risks for manufacturers, pay-TV operators and broadcasters. In particular, they face the risk of investing in middleware formats that later become obsolescent or overtaken as a viable medium by other, rival formats.

Another trend in provision of interactive services is to broadcast interactive applications within the same transport streams as conventional television services in order to enhance them. For example, a recipe application may be broadcast during a cookery programme and trigger appearance of a cue to invite users to interact with the application at certain points during the programme. Another example is an interactive

"infomercial" application that may contain information or background to a conventional television advertisement being viewed simultaneously on screen. A fundamental limitation of these current methods is that the applications are downloaded either in real-time during or seconds in advance of the broadcasts of the television programmes or advertisements they are supposed to accompany. This can result in delays to their usability while they are downloaded to platforms and, where applications are repetitively rebroadcast during programmes, wasted bandwidth also.

Aspects and Benefits of the Invention

It is recognised by the inventor that greater costs savings can be obtained for platform manufacturers and users if a proportion of a platform's flash memory is replaced by DRAM or hard disk memory, whereby the software comprising the firmware is downloaded from a broadcast or uploaded from a source across the internet periodically to and stored within DRAM or hard disk memory.

It is recognised also by the inventor that greater responsiveness to platform users can be achieved when they invoke certain, popular applications if these applications are stored, maintained and updated within DRAM throughout the platform's service life.

It is recognised also by the inventor that greater commercial certainties can be achieved for television network operators and broadcasters by storing the middleware component of firmware within volatile memory, such as DRAM, as opposed to within flash or read only memory, whereupon the type or revision of middleware can be replaced immediately during a single download to DRAM.

It is recognised also by the inventor that greater benefits to manufacturers and users can be achieved if middleware and application objects can be conditionally downloaded to platforms according to the values of parameters stored within each platform, thus allowing the appearance and functional behaviour of each platform to be customised according to its brand, model number or platform type, and/or personalised according to its users' preferences or histories of use.

It is recognised also by the inventor that greater bandwidth savings to broadcasters can be achieved if certain popular applications and middleware components are broadcast once or a small number of times during a day.

It is recognised also by the inventor that greater power savings can be obtained with consequent economic benefits to users and ecological benefits to the environment if platforms only power up from standby to download software or content that is new or whose version has changed compared to previous broadcasts.

It is recognised also by the inventor that greater bandwidth savings to broadcasters can be achieved if multiple instances of a broadcast interactive television service, each targeted to different permutations of platform types (so called "shells"), can be broadcast during a single download session.

It is recognised also by the inventor that objects that are common to multiple instances, or shells, of a broadcast interactive television service may be broadcast only once during a download session with the benefit to broadcasters of bandwidth savings.

It is also recognised by the inventor that interactive applications may be downloaded to DRAM days or hours in advance of television programmes or advertisements to cause interactive enhancements to later appear in conjunction with the said programmes and advertisements with beneficial bandwidth savings to broadcasters and improved responsiveness to users, television programme makers and advertisers.

The inventor herein discloses the means whereby by these adavantages may be realised by a single set of broadcasts that support a broad base of platform hardware, platform types and application functionalities, and in a manner where users need not be aware of, or be inconvenienced by, the downloading processes.

Embodiment of the Invention

In a preferred, broadcast embodiment of the invention (see *Figure 1*), a data centre 100 receives and aggregates interactive service applications and data from a plurality of interactive service or content providers 101, 102, 103, 104 and 105, and broadcasts said data to a plurality of television platforms 109 as a collection of data streams.

The interactive and content services provided by the data streams may include:

- an electronic programme guide (EPG), where a user may view television programme listings content on a television screen, and where such content may

include programme title, programme start and end times, descriptive text and graphical illustrations, including hypertext;

- interactive games, quizzes, polls and competitions;
- chat services, where users may compose SMS text or multimedia messages from their platform, or via some other device such as a mobile telephone, and send them directly, or via a message aggregating intermediary 103, to the data centre 100;
- information services carrying, for example, editorial, news, weather or sports results.

Referring to *Figure 2*, data centre 100 contains sub-systems that may either be co-located on one site, or may be geographically distributed across a number of sites and networked together, as described below:

- Service aggregation centre 130 which receives incoming service content and applications from external parties and processes and compiles said content and applications together to form data streams;
- Call centre 135 and/or interactive voice response system 134 for the purpose of receiving and interpreting user entitlement requests over the telephone;
- Database management system 132 for the purpose of managing and storing user and platform information on a database 133, and for communication of transactions to credit card and other payment facilities via a financial backbone 138;
- Transaction Processor 131, which receives entitlement requests from users 118, generates entitlement management packets addressed to individual users' platforms 109 and feeds these to the Data Carousel 136 for broadcast;
- Data Carousel 136 for play-out of the data streams at the required time schedules and data rates;
- A pre-mux 137 for aggregation of the streams onto a single data channel.

The data streams are fed from the pre-mux 137 to a broadcast multiplex operator 106 and then transmitted from the appropriate broadcast infrastructure 110 using the DVB

(Digital Video Broadcasting) family of public satellite (DVB-S), and/or cable (DVB-C) and/or terrestrial (DVB-T) broadcast formats to users' homes.

Referring to *Figure 3*, a typical user's platform 109 comprises a set-top-box (STB) or personal video recorder (PVR) 114 that receives television services, including the data streams from the data centre 100, from an aerial, satellite dish or cable socket 116 and connects to a television (TV) 112 for display purposes via its screen 113 to one or more users 118.

Users 118 interact with the data centre 100 by speech or by tapping key sequences via a cordless, wired or mobile telephone 111 that communicates with the data centre 100 via a data and/or telephone network 108.

Users operate a handheld remote control unit 117 for the purpose of controlling the television platform 109 as shown in *Figure 4*. Remote control 117 sends commands to platform 109 preferably by wireless means. In the preferred embodiment, remote control 117 uses an infrared transmitter 129 to send commands to platform 109 that correspond to keys as they are pressed by the user, where such keys include: platform power ON/OFF toggle 127, volume up/down 125, channel up/down 126, red/green/yellow/blue fastext keys 124, up/down/left/right cursor keys 120, OK/select key 121, 0-9 numeric keys 123 and a "service" key 128 for invoking some of the platform's interactive services described in this invention. Alternative embodiments for remote control 117 may include any device where keys are appropriately labelled to correspond to that of a TV remote control 117 such as may be achieved using a mobile or cordless telephone, a standard QWERTY keyboard, a personal digital assistant (PDA), or a touch sensitive, handheld display where portions of the display are marked with labels corresponding to remote control 117 commands.

Several other embodiments may exist for platform 109 including where the functionalities of STB or PVR 114 are integrated, or partially integrated, into the TV 112 itself. Another embodiment is where the functionality of STB or PVR 114 is performed by a domestic or office personal computer (PC) and the television's display screen 113 functionality is performed by a PC monitor. Yet another embodiment of platform 109 is where the aerial 116, STB 114, television 112, keypad 117 and screen

113 functionalities are integrated into a single handheld device, such as a cordless or mobile phone, PC notebook, palmtop computer or a personal digital assistant (PDA).

Yet a further embodiment is where platform 109 comprises a local area network (LAN) transceiver 119, as shown in *Figure 5*, where data is exchanged between the data centre 100 over the internet 108 instead of via a DVB broadcast infrastructure. Typically in such embodiments, STB or PVR 114 is connected to the internet via a local, domestic access point 139 by either wireless (such as via the WiFi or wireless IEEE 802.11a/b/g standards) or Ethernet cable means. In such cases, platform 109 may continue to receiver television channels and other services by broadcast means via infrastructure 110 and aerial 116. Alternately or additionally, platform 119 may also receive television channels and other services via the internet 108, access point 139 and LAN transceiver 119.

Platform Memory Allocations

Figure 6 describes the allocation of memory within platforms 109 in the preferred embodiment. An area of STB 114 memory within a volatile portion of dynamic random access memory (DRAM) 146, the so called "canister" area 147, is reserved exclusively for storage of downloaded object images 150 that comprise the interactive service from the data centre 100. With current memory costs, the size of the canister memory area 147 is typically 16 or 32Mbytes. Future embodiments may allocate larger memory areas to the canister 147 and may include embodiments in such platforms as for example personal video recorders (PVRs), where hard disk or magneto-optical disk (such as is widely employed by recordable CD or DVD players) memory is used for canister 147 storage in place of, or in addition to, DRAM.

Another area of memory, a so called "scratchpad" area 148, is temporarily reserved for the storage of intermediate results during particular events:

- Configuration stream 170 downloads: where the platform searches for, downloads and processes configuration stream data;
- Burst download events 171: where the platform downloads new objects into a buffer area 151 within the scratchpad area 148, processes the objects and writes them to the canister area;

- Trigger events and real-time user sessions, where the canister applications 293 execute in the foreground;
- Background downloading of the configuration 170, delta 172 or trigger 173 streams.

Outside these events, the scratchpad 148 is free for use by other non-canister resident applications such as, for example, by an MHEG-5 engine. In the preferred embodiment, the scratchpad area is composed of DRAM but in other platform embodiments, such as within a hard disk based personal video recorder (PVR), the scratchpad may be allocated hard disk memory storage, or a combination of hard disk memory and DRAM.

Portions of non-volatile, flash memory 152 are also allocated to storage of firmware and data associated with maintenance of the canister 147 and the applications 293 stored within it:

- Canister loader 140: executable code that controls the download of objects from burst streams 171, and generates a binary image 150 within the canister 147. Preferably, only a minimal "bootstrap" component 140 of the canister loader is present in flash memory 152 as firmware, and where the remaining "transient" portion 153 is loaded to volatile memory 146 at the commencement of downloads from burst events 171 and is wholly or partially erased afterwards;
- Decoders and translators 141: software modules called by the canister bootstrap loader 140 and/or bootstrap transient loader 153 to process downloaded objects. For example, the canister loader may first be required to invoke a decryption decoder to render an object to clear text. Another decoder may be subsequently invoked to decompress the clear text object. All executable objects are broadcast in a platform independent form. Preferably, the canister loader invokes a translator to convert platform independent executable objects to a form where they are directly executable by the platform's central processor unit (CPU);
- Platform adaptation layer 142: preferably objects comprising an operating system 288, which the canister applications call exclusively, is downloaded into the canister 147. The platform adaptation layer 142 is platform specific code that binds the canister operating system image 288 to the platform's native drivers and hardware;

- Platform parameters 143, such as brand, model number and physical characteristics such as CPU speed, screen resolution, size of canister and scratchpad allocations are burnt into non-volatile memory during manufacture and read only throughout a platform's service life;
- Platform Cookies 144, such as the ServiceID or entitlements (see later), carry platform configuration information which must survive power loss and consequent loss of the contents from the canister 147;
- Application Cookies 145 are files that hold certain states of an application that need to be restored the next time it is invoked. For example, a television program guide application 284 may permit a user to remove or order certain channels in his/her set-up preferences. Or, for example, a games user may have logged a name and a high score. Or, for example, an entitlement may have been downloaded that causes certain applications to allow access to all or certain additional portions of their functionalities.

Canister structure

Figure 7 shows the structure of programme software, data and service content stored within canister 147. Preferably, the canister comprises a layered software stack that is functionally identical across canisters in other platform hardware types and includes driver software 290 that control platform 109 programmable hardware (such as, for example, display controllers, sound generators, tuners and de-multiplexers) via adaptation firmware 142 that is unique to each platform hardware configuration. Preferably each canister also comprises an operating system 288 and libraries of executable code 289 that may be called via a single canister application programming interface (API) 291 by applications 293 resident within the canister 147. Preferably each application 293 may have associated with it data or content 292 which it may process. Preferably, the applications 293 include one or more game application(s) 283, an electronic programme guide (EPG) application 284 a chat application 285 and a graphical and hypertext browser 287.

Burst stream

The canister 147 is loaded with one or more objects from a burst stream 171. Typically, burst stream 171 has 15 to 120 minutes duration and a high 67-256kbit/s rate compared to the other streams in order to minimise platforms' power up times during download. *Figure 8* describes the overall sequence of objects conveyed in burst stream 171. The first object broadcast is the root object 180, which is followed by other objects 181, 182, 183, in an order such that no object is broadcast until all other object(s) that reference it, if any, have been broadcast beforehand. Finally, an "end of objects" marker 184 is broadcast to delimit the end of burst stream 171.

Objects may comprise any type of binary data, including application and/or operating system and/or driver executables in any format (i.e. including Java bytecode or script, Pascal p-code, Intent VP code, machine codes native to the platform's CPU, source codes), content of any type (i.e. including text, graphics, sound, audio and/or video clips, movies, conditional access entitlement) whether or not stored in compressed and/or encrypted form.

Figure 9 describes the structure between the objects carried in the burst stream. Canister loader 140 constructs a memory image in the canister based around a connected network of object payloads that emanate from one or more "shell" objects 181. Only one shell object 181 may be downloaded to a given platform canister, and whose identity is determined for a given platform by evaluation of logical expressions within root object 180 which are functions of platform parameters 143. A service corresponding to a subset of objects that connect directly or indirectly beneath a particular shell object 181 is referred hereafter to as a "shell service".

A shell object 181 may differ from another shell object in the sense that, while linking indirectly to content objects 183 that cause a service to appear identical to users, it may link directly or indirectly to object payloads that contain a different operating system, e.g. one shell links objects into the canister 147 comprising Multimedia Home Platform middleware while a second shell object, for example, might link in a proprietary operating system such as PSOS or Intent, while a third, for example, might layer MHP over a proprietary operating system. As industry acceptance of application programming interfaces (APIs) evolves, both manufacturers and broadcasters may seek to alter the middleware in order to adapt to changing industry

needs. The author's invention, whereby full canisters of firmware and applications are regularly broadcast and updated within memory, brings major benefits to users, manufacturers and broadcasters because it allows such systems to be more easily and quickly updated compared to a conventional approach of burning middleware or other firmware into flash memory.

Further, visible differences may exist between two shell services running on different platforms but where no or limited structural or programming differences exist between the groups of objects in their respective canisters. For example, a shell service may differ only in terms of its "skin" (i.e. the style, backgrounds or appearance of the service to the user) but be identical in terms of the functionality and content it displays. This is especially important for manufacturers who need to show, at minimum, differentiation in terms of aesthetics and appearance from similar services displayed on other manufacturers' products.

A single burst stream 171 may additionally support a diversity of multiple, different platform types. For example, the service operator may support two shell services: one serving set-top-box platforms, and another serving, say, personal video recorders. Whereas, both services may convey essentially the same content, certain executable objects may be required for one platform but not the other – and vice versa. Similarly, certain content objects may be designated for storage on one platform but may not, possibly for reasons of size, be designated for use on other platforms.

Platform configuration

Figure 10 illustrates how the data centre 100 broadcasts a number of data streams throughout a day, keeping the canister 147 up to date and filled with applications and other objects.

Platforms 109 periodically power up their tuner and data receiver stages at the beginnings of burst streams 171 to reload their canisters. Timing and location information to allow platforms to determine when and on which transport stream the next burst event will occur are carried in configuration streams 170. Turning to *Figure 11*, configuration stream 170 is a perpetual, repeating (typically every few seconds) low bit rate stream, typically less than 2kbit/s, carrying service configuration and system data such as time, the locations of all data centre 100 streams,

ConfigurationTable 162, and schedules and locations within transport streams of all forthcoming burst events 171 within the near future, *BurstEventTable* 163, typically within the next 24 hours. Importantly, burst events may be broadcast for different services. For example it may be possible to receive channels on two DTT networks near the French-German border, as Figure 11 illustrates. In which case different service identities 207 may be broadcast to support each.

Figure 12 describes a perpetual loop whereby the canister loader 140 periodically reads the configuration stream to determine when and on which transport stream the next burst event will occur. While the platform is in standby mode (1-1), the canister loader stays within a loop where, every 2 hours (1-9), it sends a "power on" command to the programmable tuner, de-multiplexer component (1-10) and next acquires data from the configuration stream according to the *LoadConfiguration* process (1-11) (see later) and then powers down the programmable tuner and de-multiplexer components (1-12). If the platform is powered up, the canister loader determines whether the current time is within a user permitted download window (1-2). This window may be changed by the user via a set-up menu but, by default, is set to between 2am and 5am in the early morning. Given that a channel change is likely to be required in order to acquire the configuration, the canister loader attempts to minimise disruption to any viewing activity by attempting the acquisition only after a sustained period of user inactivity (1-3, 1-4, 1-5). The inactivity period is 4 hours by default, but can be changed by the user via a setup menu. At which time, the canister loader overlays a message onto a portion of the screen warning that a configuration will occur if the user does not acknowledge by hitting the "OK" key on his/her remote control (1-6). If no acknowledgement is given (1-7), the canister loader saves the platform's current tuned channel settings to memory (1-15), initiates the *LoadConfiguration* process (1-14) and then restores channel tuning to the current channel (1-13). If an acknowledgement is given, the canister loader aborts its attempt to download, resets an inactivity countdown timer (1-8) and returns to the loop.

Figure 13 describes the process, *LoadConfiguration*, whereby canister loader 140 determines the parameters associated with the next burst event. Canister loader searches the transport streams for a valid configuration stream (2-2). If a configuration stream is not found (2-3), the canister loader displays a diagnostic message to screen (2-9), otherwise it loads the current time and date from the

configuration stream and sets the platform real time clock accordingly (2-9). Next, canister loader 140 looks in the non-volatile memory area for a platform cookie 144, *ServiceID*, that identifies which service or services, the platform has previously been configured by the user to receive and load into canister 147 (2-5). If the *ServiceID* cookie exists then canister loader 140 looks up the identity of the next burst event, *BurstID*, that corresponds to *ServiceID* (2-10).

If the *ServiceID* cookie does not exist (i.e. because the platform is new), or its value cannot be found in the *BurstEvent* table, the user is invited to select via a pop-up on-screen menu a service from those featured within the *BurstEvent* table.

Given that a plurality of consecutive burst events 171 may carry identical objects (i.e. because the service has not been updated between each) it is undesirable for the platform 109 to waste power by waking from standby to download the same objects multiple times. A means whereby the canister loader 140 avoids downloading the same data already present in the canister is desirable, and is achieved by adding to the *BurstEvent* table a service attribute, *ServiceVersion*, that is unique for each set of objects of a given service identity 207 that is carried within a burst event. Each time canister loader 140 loads a shell of objects to the canister, it writes also two variables, *CanisterServiceVersion* and *CanisterShellVersion*, into the canister 147 that correspond to unique versions for all objects within the service and the loaded shell respectively (2-14).

Hence, the canister loader determines first whether *ServiceVersion* and *CanisterServiceVersion* match (2-6, 2-7). If they do, then there is no requirement to download burst event 171 corresponding to *BurstID* and so the process for configuring the next burst event ends (2-15). Second, the canister loader determines whether *ShellVersion* and *CanisterShellVersion* match (2-8, 2-12). Finally, if no match is found between either the service or the shell versions, the Canister loader writes the *BurstID*, *NextBurstTime* and *NextBurstTransportID* into the platform cookie area (2-14), so that the platform is now properly configured with the next burst event's time and location, whereupon the *LoadConfiguration* process ends (2-15).

Delta stream

Burst stream events occur only periodically during a day. A repetitively broadcast, carousel stream of updated objects may be broadcast between burst events, referred to as the delta stream 172.

Figure 14 shows the structure of the delta stream, and illustrates a case where newer versions of certain objects 185 have become available for broadcast since commencement of the "B" burst event 171 described in *Figure 10* and *Figure 9*. Broadcast of these updated objects 185 is delimited by a beginning of carousel marker 186, followed by the updated objects 185 and terminated with an end of carousel marker 187. The broadcast is repeated on delta stream 172 normally for at least 24 hours, or until a burst event 171 that includes the updated object versions (event "C" in *Figure 10*), or newer versions still, has ended. An example is a modified news object to reflect a news bulletin or an updated television listings object to reflect a schedule change. Delta stream 172 has a low bit rate, typically 13-32kbit/s, compared to the burst stream.

Delta stream 172 may be processed by the platform either transparently or manually during a user session. In the manual case, a user invokes a canister application 293 that may cause the platform 109 to re-tune to receive and process the delta stream 172 in real-time. For example, a user may select, say, a "listings update" feature within an IPG application 284 to cause it to invoke the canister loader 140 for the purpose of updating listings content objects 281.

Figure 15 describes the process, *LoadDelta*, whereby canister loader 140 tunes to the delta stream 172 to parse and replace old canister objects with newer ones found in the delta stream 172. First, the canister loader saves the programmable tuner's existing settings (3-2), reads *ConfigurationTable* to determine on which transport stream and data stream identity (packet ID) the delta stream is to be found (3-3) and then sends a command to the programmable tuner and de-multiplexer to filter on the delta stream (3-4). The canister then waits until a beginning of objects marker 186 is received before invoking process *LoadDeltaObject* to process each object (3-6). *LoadDeltaObject* is described in *Figure 16* and is essentially a simplified form of *LoadBurstObject*, except that old object images are erased from the canister (4-15) as

each new image is added preferably using a "double buffer" method where, in order to guard against data loss, the old image version is unlinked and deleted only when the new image has been fully written into the canister and linked to other objects.

Optionally, either the entirety or a subset of the canister may also be broadcast continuously within the delta stream. This is useful to users who are powering up their platforms from a powered off state, or whose platforms are new and being used for the first time, and where users cannot wait until the next Burst event and need quickly to reload their canisters so that the platform becomes useable. In such cases the user manually initiates the download process *LoadDelta*, as illustrated via a setup menu and where the canister loader processes objects in real-time from the delta stream 172 in place of the burst stream 170.

Trigger stream

The canister 147 may provide interactive functionalities that complement or enhance viewing of conventional television channel services. During such services, small trigger data objects are broadcast in the same transport stream or multiplex as the television channel event so as to activate specific portions of an application that is already stored in the canister 147.

The triggering method is described as follows: *Figure 17* shows a conventional television service channel displayed 220 on screen 113 upon which a canister 147 resident application is caused to display a first overlay 270 over channel display 220, as shown in *Figure 18*, responsive to a triggering signal generated by the television channel playout facility 107. The video 220 may be scaled down, simultaneous to the appearance of overlay 270, to any size within the screen so that the video and pop-up overlay 270 can appear simultaneously on screen 113 without overlap between them. Overlay 270 may be opaque and obliterate the portion of video 220 it covers or, alternatively, it may be translucent so that the covered video 220 is partially visible through overlay 270. The overlay 270 may be any size or shape, such as circular, oval or rectangular, and be positioned anywhere on the screen 113. The overlay may contain an instruction telling the user which keys to press on his/her remote control 117 to cause second and subsequent overlays 271, such as described by *Figure 19*, to appear. Generally, all overlays may optionally contain interactive display

components such a text labels 272, static or moving graphics 274 and a highlighting effect 273 so as to illustrate an area of focus to the user within said overlay, where a user may press a key (the "OK" key 121 in the example overlay 271 illustrated) on remote control 117 to select a process for execution within the interactive application 293 associated with said highlight 273.

Figure 20 shows the process, *TriggerStreamHandler*, employed to cause applications 293 to overlay the screen during conventional television viewing and is described as follows. First, *TriggerStreamHandler* process is started when the platform 109 is powered up from power off or standby mode and runs continuously in the background monitoring which television channel the platform is tuned to (5-1). Each time a channel change occurs (5-2), the new channels service and transport IDs are acquired (5-3) and looked up (5-4) in *ConfigurationTable* to determine whether an identifier for the trigger packet stream PID can be found (5-5). Assuming a trigger PID exists, *TriggerStreamHandler* reads the next trigger object into scratchpad 148 (5-7) and checks that is not corrupted and authentic before decrypting (5-8, 5-9). Trigger objects have identical structures to burst or delta objects 200, with each potentially carrying multiple payloads 205. The payload expressions 204, if any, are evaluated to identify which payload is to be retained (5-11, 5-12). Unwanted payloads are erased. Trigger payloads 280 are generally small and of the structure shown in *Figure 21* where, typically a payload contains only a single decoder/translator 281 known as a "trigger handler". The trigger handler is executed (5-15) and may be accompanied by a binary image 282 containing information linking the identity of a particular television service channel 295 to a particular canister application 296 in order to convey a message "token" 297 at a particular time or upon the occurrence of particular timestamp (carried within the tuned transport stream) 298. The trigger handler 281 starts the identified application 293 if is not running and uses an inter-process communication pipe to communicate token 297 to the identified application 293 which, in turn, causes overlays 270 and 271 to appear on screen 113.

Transaction stream

An individual platform addressable, "Transaction" stream 174, is broadcast continuously throughout the day. The Transaction stream contains feature

entitlement information that is addressed to individual platforms, the process for which is described in detail later.

Objects and payloads

A shell object 181 links to one or more operating system objects and one or more application objects 182, each of which in turn may link to other objects 183. An object may be linked to by multiple objects, but is broadcast only once during a burst event 171 to conserve bandwidth. The canister loader writes all the objects carried on the burst stream into a buffer area 151 within the scratchpad where they are processed, before writing to the canister the linked root, tree and branch of objects applicable to each – as described below.

All objects are tagged with an identity, *ServiceID*, 207 of the service to which they belong. For example two burst events 171 may co-exist on the same stream (e.g. a French service and a German one). All objects comprise the structure described in *Figure 22*. Each object also has an *Identity* 201 for the purpose of being linked to by other objects. All objects have an integer *Version* 202 associated with them. Objects which have been updated since a previous broadcast (for example, an object containing TV listings may have been updated since its broadcast in an earlier burst event), carry incremented *Version* integers compared to the respective previous values. An object may carry a signature and/or checksum 203 to authenticate its integrity. An object may carry multiple payloads 205. Payloads are embedded within a case statement so that, normally, only one payload is chosen by the Canister loader for loading into the canister according to whether its associated logical expression 204 is true. The canister loader contains an interpreter with which it evaluates on the fly logical case expressions. Expressions are logical functions of operators, platform parameters 143 and constants. Operators may include arithmetic operators (plus, minus, multiply and divide), logical (NOT, AND, OR), inequalities (<=, >=, <, >, !=, ==), string operators (LEN, FIND), wildcard (*, ?), memory peek and array pointer operators. Platform parameters and constants may be of type character, short and long integer, float, string, time/date and boolean. Expressions containing a reference to a platform parameter 143 that is not satisfied are evaluated as false.

Downloading objects from the burst stream

Figure 23 describes a perpetual loop whereby the canister loader 140 periodically reads the burst stream 171 in order to regenerate the canister 147. While the platform is in standby power mode (6-1), the canister loader reads a real-time clock to determine whether the time is within a guard time (15s) of the next burst event (6-8), as previously determined by the *LoadConfiguration* process. If so, the canister loader sends a "power on" command to the programmable tuner, de-multiplexer components (6-9) and initiates the canister download *LoadBurst* process (6-10) (see later), followed by initiation of a *LoadConfiguration* process (6-11) in order to acquire a configuration for the next burst event, finally followed by a "power down" command to the programmable tuner, de-multiplexer components (6-12) before repeating the loop.

If the platform is powered up (6-1), the canister loader determines whether the current time is within a user permitted download window (6-2) – where the window is the same as described earlier for loading of configuration parameters. Given that a channel change is likely to be required in order to acquire the burst event objects, the canister loader attempts to minimise disruption to the viewer by attempting the acquisition only after a sustained period of user inactivity (6-3, 6-4). This inactivity period is 4 hours by default, but can be changed by the user via a setup menu. At which time, the canister loader overlays a message onto a portion of the screen warning that a configuration will occur if the user does not acknowledge by hitting the "OK" key on his/her remote control (6-5). If no acknowledgement is given (6-6), the canister loader saves the platform's current tuned channel settings to memory, initiates the *LoadBurst* process (6-15), followed by the *LoadConfiguration* process (6-14) and then restores channel tuning to the current channel (6-13). If an acknowledgement is given (6-6), the canister loader aborts its attempt to download, resets an inactivity countdown timer (6-7) and returns to the loop.

Loading the service and remaining object(s), if any

Figure 24 describes the process, *LoadBurst*, used by the canister loader for parsing all objects broadcast in the burst stream. The process starts with forming an empty table in memory (7-2), *UnsatisfiedLinks*, that accumulates link references during the load,

then by loading the first, root object 180 into the scratchpad area 148. Preferably the bulk of the canister loader's functionality is contained within a downloaded, "transient" image 212 carried within payload(s) 205 of root object 180. The first object, the root object 180 always has a null (0) *Identity* 201 and is downloaded by the bootstrap portion of the canister loader 140. The root object's payload expression(s) 204 is(are) evaluated to determine which payload is to be used. Preferably, the root payload contains the remaining "transient" component 153 of the canister loader in a device independent form. If the payload's binary image is in a compressed or device independent form, the payload contains references 210 to one or more decoders and/or translators 141 in firmware 152 which are invoked in sequence to render the image in executable form. Finally the image is executed to invoke the canister loader's full functionality as described below.

Figure 25 illustrates the process *LoadBurstObject* (7-4, 8-1) used by the canister loader to parse each object, including the root object 180 and is further described as follows. Unless the object is the root object 180 (8-3), the object's *Identity* 201 is looked up in *UnsatisfiedLinks* to determine whether it has been referenced by a preceding object. If the object's *Identity* is present in *UnsatisfiedLinks* (8-4) then the canister loader determines whether an object with the same identity and *Version* has already been loaded into the canister (8-5, 8-6). If not, the raw object data is written to an area in scratchpad memory (8-8) where its signature and/or checksum is checked to verify authenticity and that it is error free (8-9). Assuming the object is authentic and error free (8-10), it is decrypted to clear text (8-11) whereupon, in cases where the object comprises multiple payloads (8-12), payload expression(s) are evaluated to determine which payload is to be processed and loaded (8-13). The canister loader then executes the coders and translators referenced in the payload's header (see Figure 26) to process the payload's binary image sequentially (8-14, 8-15, 8-16), and then writes it to memory within the canister (8-17). Next, the canister loader reads each object link, if any, referenced by the payload and checks to determine whether each reference object has already been loaded to the canister (8-18, 8-22). If a referenced object has not been loaded, its identity is added to the *UnsatisfiedLinks* table (8-23, 8-24). Finally, the loaded object's entry in *UnsatisfiedLinks* is removed (8-20).

The parsing process continues for all objects until a broadcast end of marker 184 is reached (7-3), whereupon the transient canister loader component 153 or a substantial proportion of it, if any, is erased (7-6) from the scratchpad area 148. A proportion of the transient loader may persist in scratchpad or alternatively may be located in the canister for the purpose of processing delta streams 172 and trigger streams 173.

Service Entitlement

Because object downloading normally only occurs at scheduled times, as opposed to in real-time during a session when a user may seek entitlement to use or view them, all objects belonging to a shell are loaded to the canister in advance of a session and irrespective of whether the user plans, or is entitled, to "access" them by using the functionalities or viewing the content they contain. Consequently, a method is desired whereby certain functionalities within applications can be activated or "entitled" in real time by a user. This is described below.

Figure 17 illustrates a viewing screen 113 where a user 118 watches video 220 as either a telecast television channel or as a video segment that is being played out from storage local to the platform, and where the user has selected using the "Service" key 128 on remote control 117 the canister's interactive services to which he/she does not currently possess an access entitlement. A "call to action" overlay 221 appears on the screen 220, as shown in *Figure 27*, to inform the user of the entitlement proposition and to give instructions on what to do to obtain entitlement. The video 220 may be scaled down to any size within the screen 113 so that video 220 and pop-up overlay 221 can appear simultaneously on screen 113 without overlap between them. If the video is played out from local storage, it may be paused while the overlay is displayed. Overlay 221 may be opaque and obliterate the portion of the video 220 it overlays or, alternatively, it may be translucent so that the covered video 220 is partially visible through the overlay 221 which may be any size or shape, such as circular, oval or rectangular, and be positioned anywhere on the screen 113.

Alternatively, only certain objects associated with a shell may require verification whether a user is entitled to access. For example, the first level of an interactive game application object may be accessed by all users irrespective of whether they have an access entitlement. However, subsequent levels within the game may require the user

to have taken out, say, an annual subscription to play all games or some other form of entitlement, such as pay per play of an individual game, in order to play the game's higher levels. An electronic programme guide application 284 where certain of its features or certain of its content 281 requires some form of paid entitlement to use or view them is another example.

Preferably, the platform contains or is connected to a modem 115 so that the user may select a desired entitlement option without placing a manual telephone call by using the cursor keys 170 on his/her remote control 117 to move a highlighting effect from one to another of a plurality of text areas or to areas associated with text labels within the pop-up. The highlighting effect signifies user focus on the screen and may be a cursor or some other icon marking the text, or it may be a changed border surrounding the area, or a changed area background or pattern or a changed font style, or a flashing effect, or some combination of the aforementioned.

In the preferred embodiment a user can, while watching full screen telecast television channel or while playing a television programme stored within the platform 220, and cause a menu of the canister's applications to be displayed on the screen, as illustrated in *Figure 28*, by pressing the "Service" key 128 on his/her remote control 117. The screen displays a plurality of applications (e.g. program guide 227, games 228, shopping 229) within a menu area 231, where one application is highlighted 228. A second area of the screen 226, that does not overlap with the menu area 231, gives descriptive information concerning the highlighted option 228. The user may press the cursor keys 120 to move the highlighting to another option within the menu area, whereupon the contents of the notes area 226 change in response to the newly highlighted option. A scroll arrow 230 appears when a user can navigate the highlight in the direction of the arrow 230 to a menu option which is currently not displayed or is outside the menu area 231. Simultaneous to the menu and notes areas, a third non-overlapping area 224 displays a reduced scale video 220 corresponding to the telecast channel or television previously viewed in full screen. A user may further select an application 228 by pressing the "OK" key 121. If the user is not currently entitled to access the applications in the canister corresponding to the highlighted option, the notes area 226 is replaced by instructions giving entitlement options 241 as shown in *Figure 29*. Preferably the options correspond to areas 242 which a user may navigate

highlighting 243 to and select by pressing the "OK" key 121. In such instance, the highlighted application 228 in the menu bar 231 is highlighted differently 243 after it is selected to signify that focus has switched to 243 one of a second series of options 242 in the notes area 241. Alternatively, if the user is entitled to access the highlighted option 228, the application is invoked or a next menu of applications corresponding to the selected options is displayed as shown in *Figure 30*.

Any executed object, either at the onset of or at some later point during its execution, can cause a call to action pop-up to be displayed on the screen. For example, *Figure 31* shows a game which has been partially played, where the user has met some criterion that permits him/her to access other functionality within the application. The user is invited to select the new functionality by pressing the "▶" (right arrow) key on the handheld remote control unit, causing the screen of *Figure 32* to be displayed. *Figure 33* describes the case where a modem is fitted to the platform and where the user has met some criterion within an application, as previously described in *Figure 31*, and where the user can purchase an entitlement by making an impulse response. In this case, when the "▶" (right arrow) key 120 on the handheld remote control unit 117 is pressed, the application causes an overlay 260 to be displayed that contains user instructions and a plurality of entitlement options including the option to back out or not to purchase an entitlement. The user navigates the highlighting to his/her desired option and presses the "OK" key 121. A modem 115 connection is then established with the service provider, whereupon an entitlement message is broadcast to and received by the platform 109, which causes the platform to execute the application, or the next level of the application as shown in *Figure 34*.

A process of broadcasting positive and negative entitlement messages is used to control individual user access to the platforms. This process is illustrated in *Figure 35* and is described as follows. An application causes a call to action pop-up to be displayed to the screen (9-5) where it has determined (9-3) that the user is selecting pay functionality (9-2), where (i) a user had attempted to use a part of that application's functionality to which the user is not currently entitled (e.g. selecting a pay game, or selecting a pay feature within the program guide), or (ii) the user's interaction history with the application has met some criterion embedded within the application's functionality (e.g. exceeding a minimum game score).

The application searches within the Applications cookies region in flash memory for the correct entitlement message name *EntitlementName* (9-3). If one exists and its contents are valid (see later) then access is granted to the pay portion of the application (9-4). A call to action message is displayed on screen (9-5) for the user if valid entitlement message contents are not found. The user decides whether (9-10) to request entitlement or whether to abandon the request and return to the non-pay components of the application's functionality (9-11). The application determines whether a modem is connected (9-9) and, if so, whether it can make a call (9-7). If so, an automated method *AutoEntitlement* is used (9-6).

If a modem does not exist, or a connection cannot be established, then the applications uses the method *ManualEntitlement* (9-8, 10-1), see *Figure 36*, which is described as follows: the applications first freezes any on screen television video and stores current channel tuning settings (10-2) (so that these may be restored later). The application then sends a command to the programmable tuner/de-multiplexer causing the platform to tune to the multiplex carrying the transaction stream, de-multiplex and filter it according to its packet ID (PID) and transport stream identity as recovered from *ConfigurationTable* (10-3). If the transport stream is not found (10-4), (e.g. due to a signal coverage problem), then a diagnostic message (10-10) is displayed to screen and the process ends (10-20). Otherwise the application looks up to determine whether the platform has a *BroadcastAddress* already assigned to it (10-5). If no *BroadcastAddress* has been previously assigned, a random number is generated for *BroadcastAddress* which is written to non-volatile platform memory (10-11). A checksum, typically an additional group of 3 to 5 digits, is added when displaying *BroadcastAddress* on screen to assist recognition by the call centre 135 of user misquotes. *BroadcastAddress* is represented preferably for display purposes to base 33 as uppercase alphabetic and numeric characters, where 3 characters (O which looks like zero, 1 which looks like I, Q which looks like zero) are not used to reduce user recognition errors, in groups of 3 to 5. A call to action 260 is displayed to screen (10-6) advising the user of the entitlement proposition describing: service features and benefits, price, telephone number to call to activate the entitlement and *BroadcastAddress* to quote. Then the application filters for an entitlement message broadcast to it which, when downloaded, has the identity *EntitlementName* (see later).

The user dials the call centre 135 using the telephone number displayed (10-7). The call centre's customer service representative (CSR) answers the user's call (10-8). The user gives the CSR his/her contact and payment details (10-9). The user reads the *BroadcastAddress* off the screen to the CSR (10-12). The CSR checks within his/her database to determine the same BroadcastAddress has been previously assigned to another user (10-13). If so, the CSR asks the user to key into the remote control a certain key sequence (10-22) that causes the platform application to generate a new *BroadcastAddress* and redisplay it to screen within the call-to-action overlay (10-21), whereupon the CSR asks the user to read it out again so that the process of checking its uniqueness can be repeated. When the *BroadcastAddress* is verified by the CSR to be valid and unique (10-13), the CSR causes an entitlement message to be broadcast to the address *BroadcastAddress* platform address (10-14). The entitlement message comprises a packet containing the following information:

- *BroadcastAddress [, ...BroadcastAddressM]:* where *[...]* contains optional broadcast addresses for other platforms that are to receive and process the same message;
- *EntitlementName* is the name of the entitlement message, for platform file system storage and later retrieval purposes by applications;
- *Functionality1 [, ...FunctionalityN]:* where *Functionality1* is a unique identifier for a particular functionality within an application or group of applications, to which access is to be entitled, and where *[...]* contains optional references to other functionalities whose access is to be granted by the same message;
- *StartTimeDate:ExpirationTimeDate:* where *StartTimeDate* is the time and date from which access shall commence, and *ExpirationTimeDate* is the time and date from which access shall expire;
- *UseCount* is the maximum permitted number of cumulative uses of the functionality within the application or group of applications (0 implies unlimited);
- *UseTime* is the maximum permitted cumulative usage time of the functionality within the application or group of applications (0 implies unlimited).

The above parameters allow entitlement messages to be used to control access on either on the basis of elapsed time, number of play sessions or duration of play

sessions, or combinations thereof. It will seen to those skilled in the art that additional flexibility may be obtained for the user entitlement process through additional parameters in entitlement messages, such as prepay tokens.

The platform downloads the entitlement message and stores it within its filing system in a non-volatile memory area as *EntitlementName* (10-15). As soon as the platform application recognises the existence within the platform filing system of a new message, *EntitlementName*, it opens the message to read the entitlement parameters. Assuming these are valid, a message acknowledging the entitlement is displayed to screen (10-16), and the CSR asks the user to acknowledge that he/she has received the entitlement on screen (10-18). If the user acknowledges the CSR thanks the user and both hang up. If the user does not acknowledge, the CSR attempts to resolve the problem with the user which, if unsuccessful, may result in the CSR scheduling a disentitlement message to be sent during one or more of the next data stream bursts (10-23).

Set-top-box hardware configuration

Figure 37 shows the internal functional elements of a typical digital STB 114, designed to receive and decode DVB TV transmissions. This is typical of the functionality with which the canister loader interoperates in the consumer TV market. This comprises a CPU 303 coupled to volatile DRAM 146 and non-volatile (flash) memory 152.

If the STB has PVR capabilities, there will be some form of bulk storage interface present 304. This would typically be an ATAPI or SCSI hard disk interface, but any popular bulk data storage interface standard may be implemented.

The STB contains a programmable tuner 300, which is connected to receive DVB-T broadcasts via an aerial 116. Additionally, the tuner may be designed to receive cable and satellite transmissions. By means of the internal data bus the canister loader 140 application software can program tuner 300 and de-multiplexer 301 to receive any MPEG2 transport stream (channel) present at aerial 116, including the streams (channel) carrying the data centre's 100 transmissions.

The tuned transport stream is applied to a de-multiplexer 301, where elementary audio, video and data streams can be extracted.

Video data streams are applied to the MPEG2 video decoder 302. The output of this decoder is then combined with the on screen display OSD to provide the video signal to the TV 112. The OSD is responsible for displaying all graphical elements of the canister applications. The video mix and scale function are capable of scaling the decoder video in order to present a reduced size live video display anywhere on TV screen 113.

Many of the functional elements described in *Figure 37* may be combined on a single large-scale integration (LSI) silicon component. In the case of an Integrated Digital TV (IDTV) all the functions described in *Figure 37* are resident within the TV chassis.

Figures

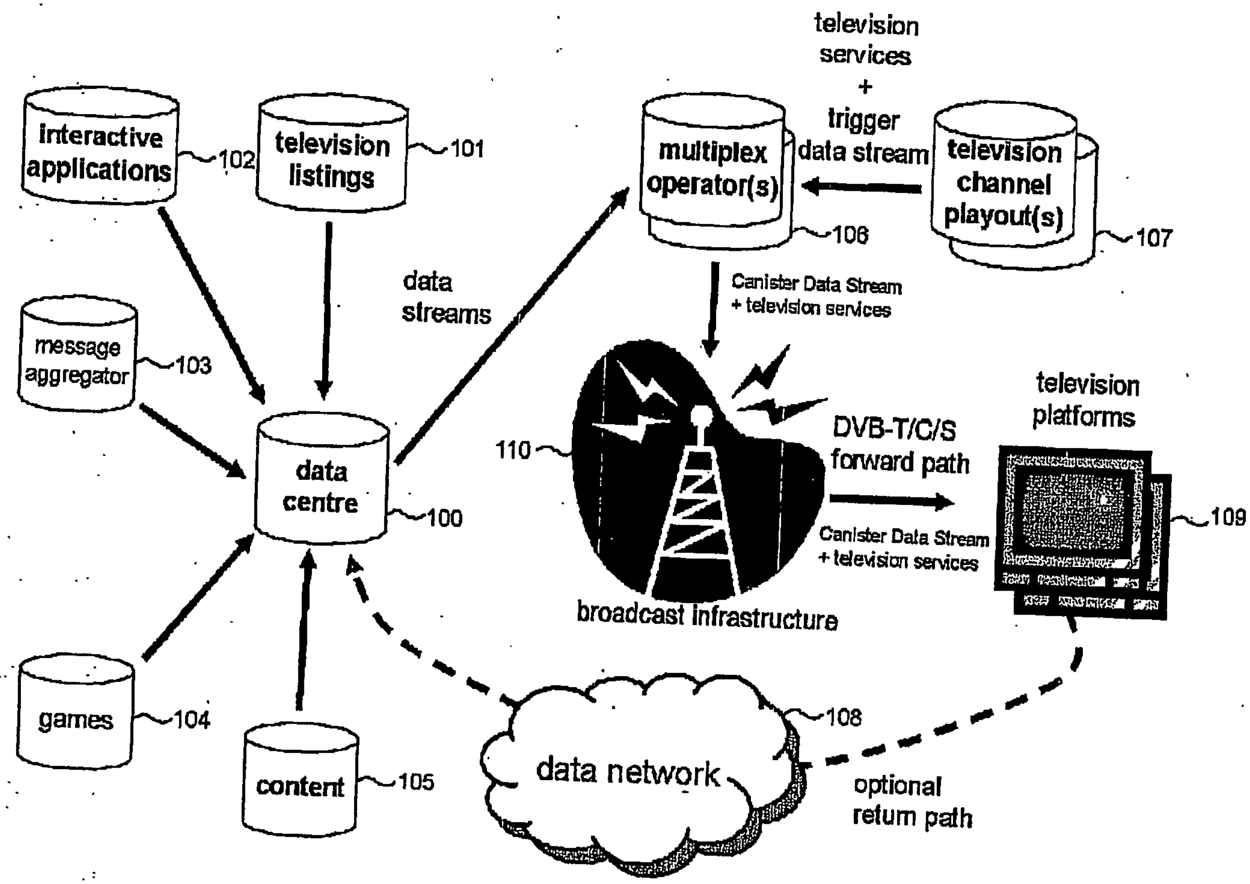

Figure 1: Overall diagram of a broadcast television system supporting an improved interactive platform.

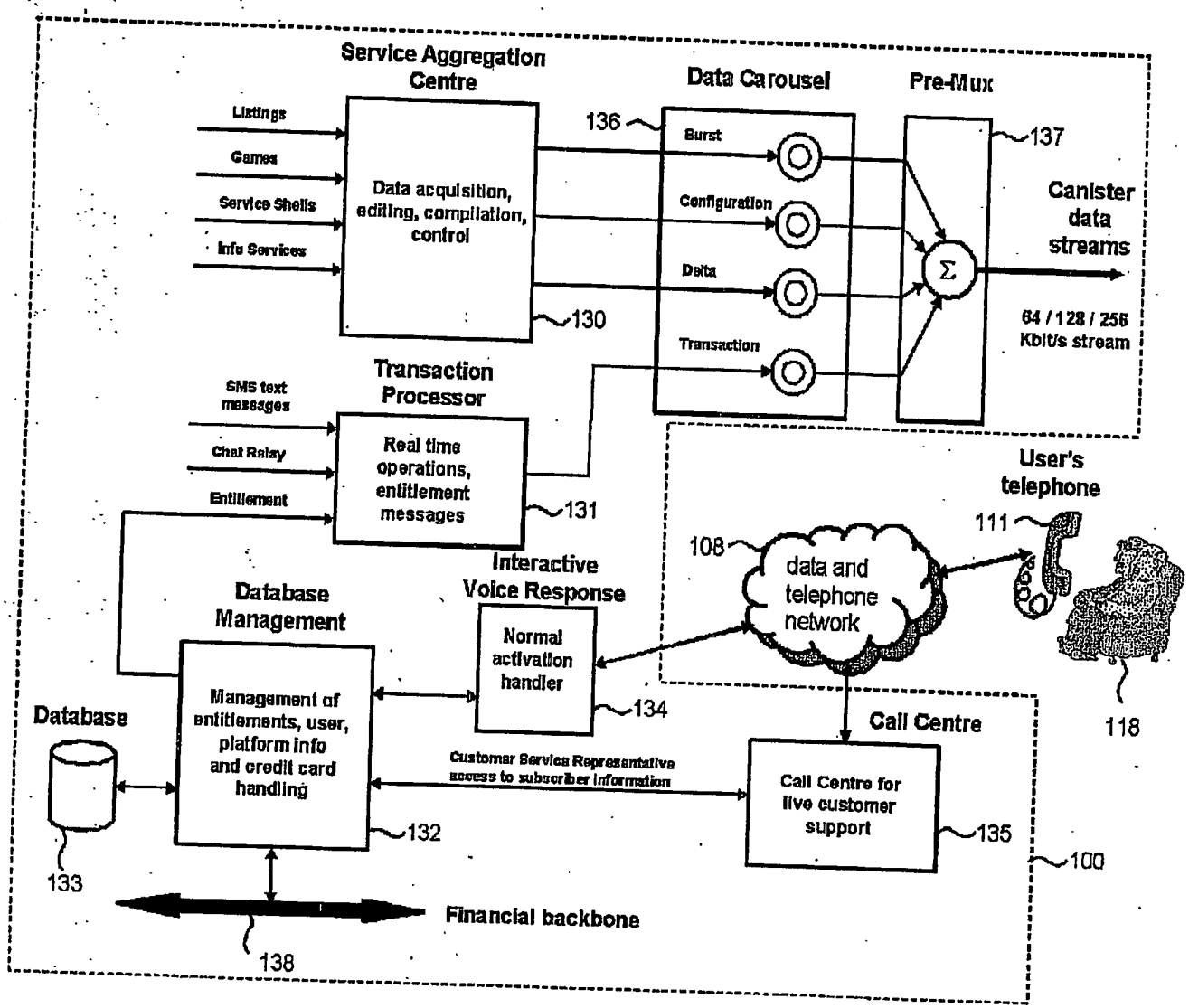

Figure 2: Data Centre and interfaces with outside entities

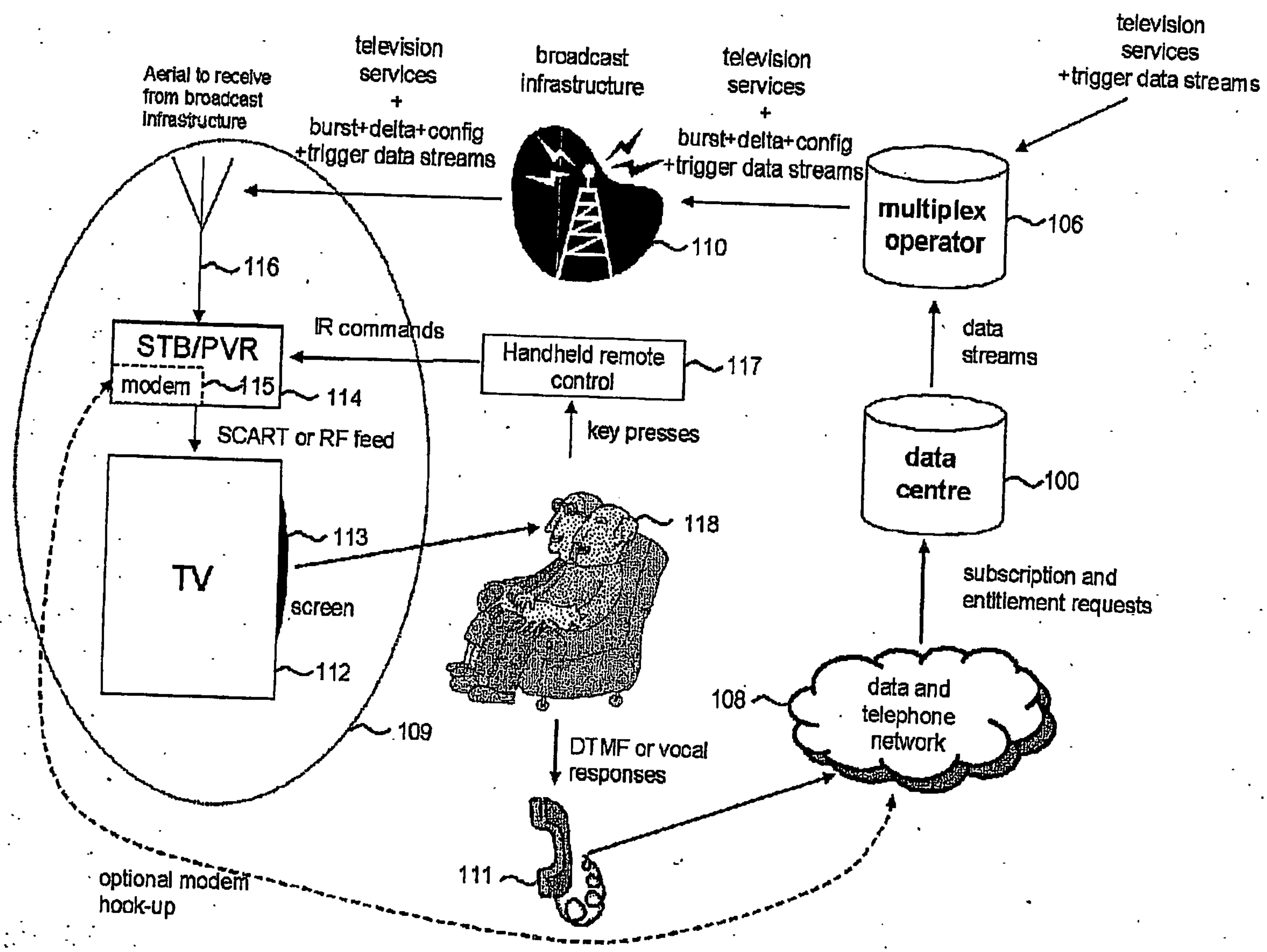

Figure 3: Broadcast fed television system and interaction with external entities.

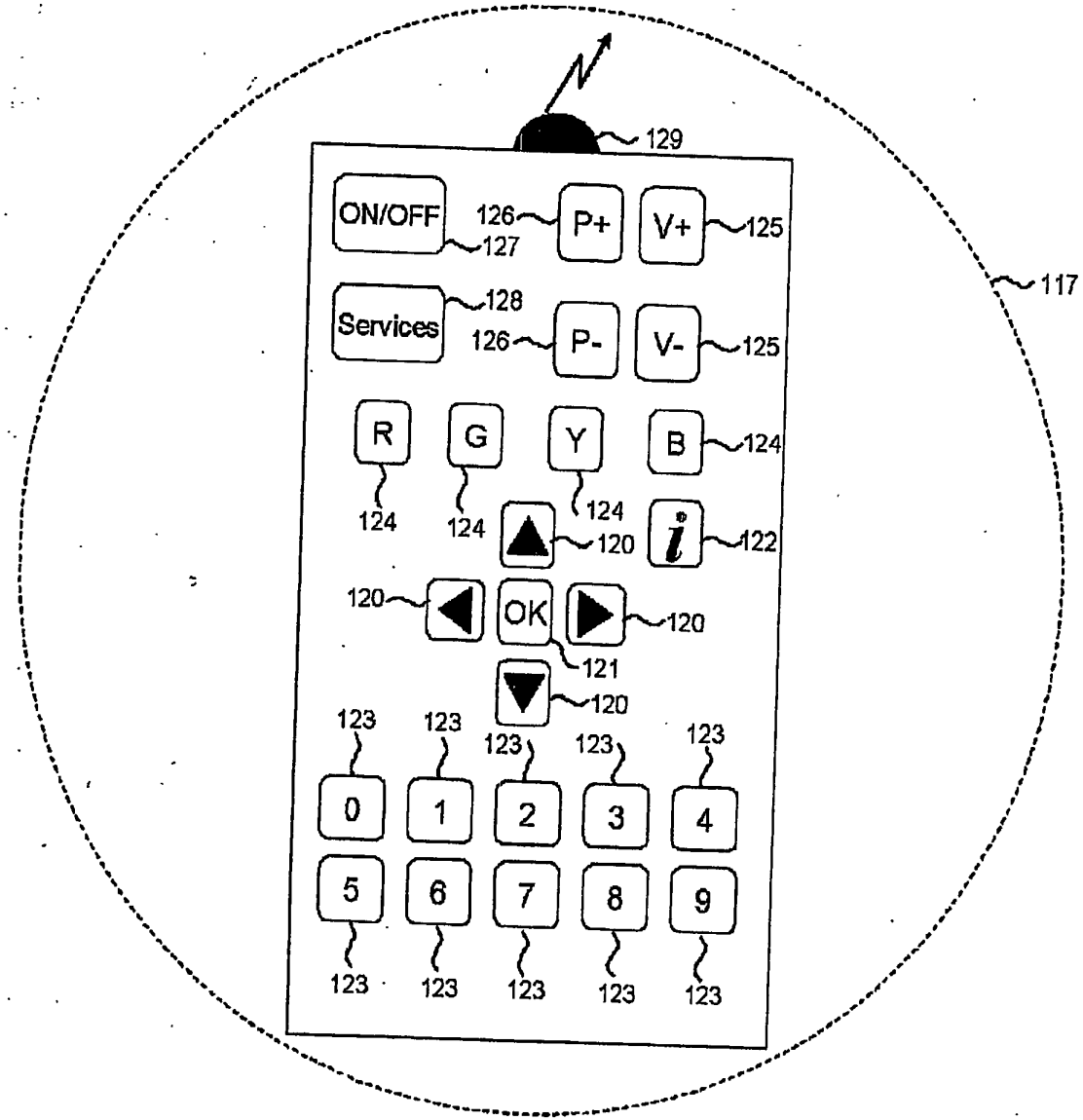

Figure 4: Diagram of handheld remote control

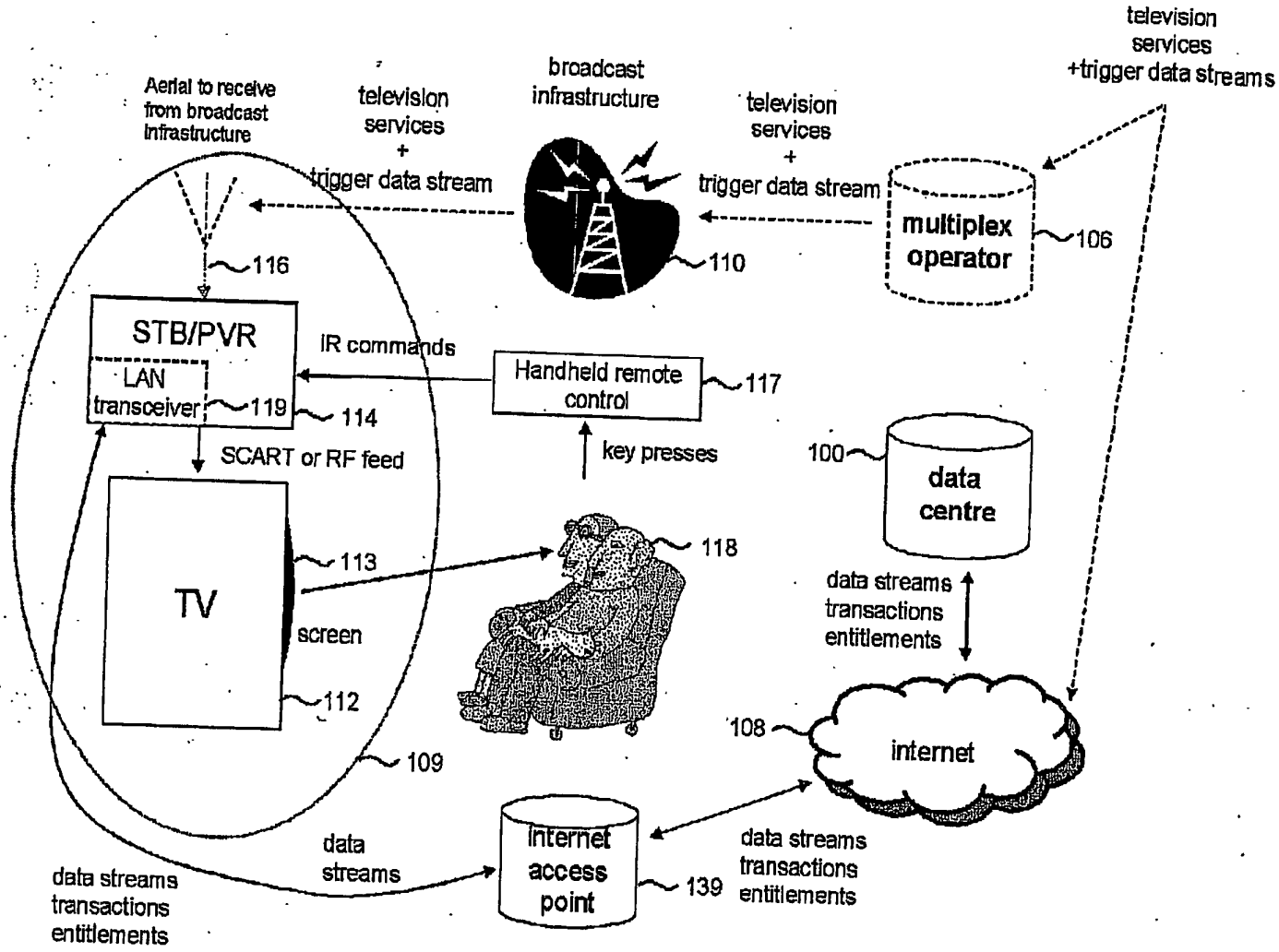

Figure 5: Internet fed television system and interaction with external entities.

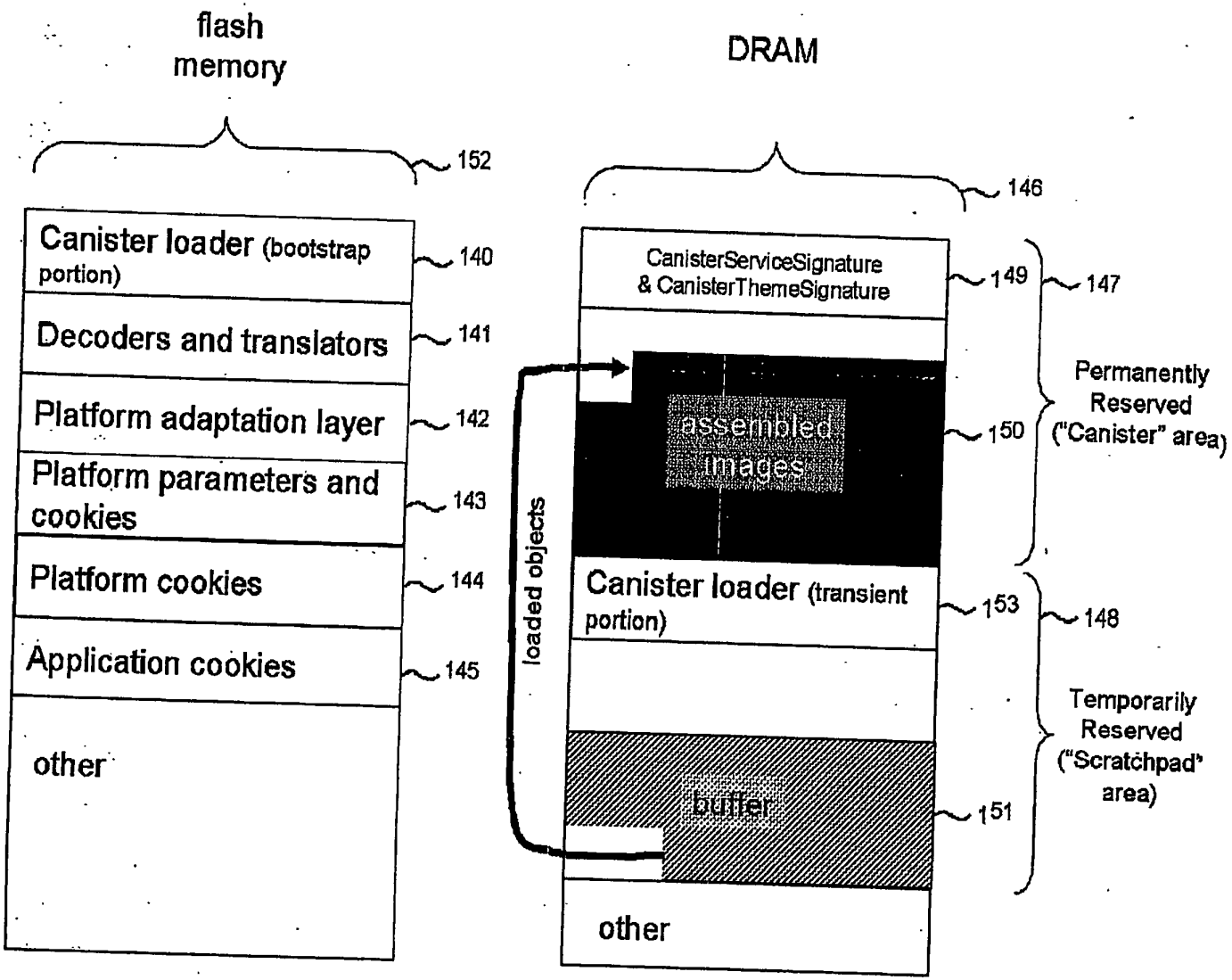

Figure 6: Platform Memory Allocation

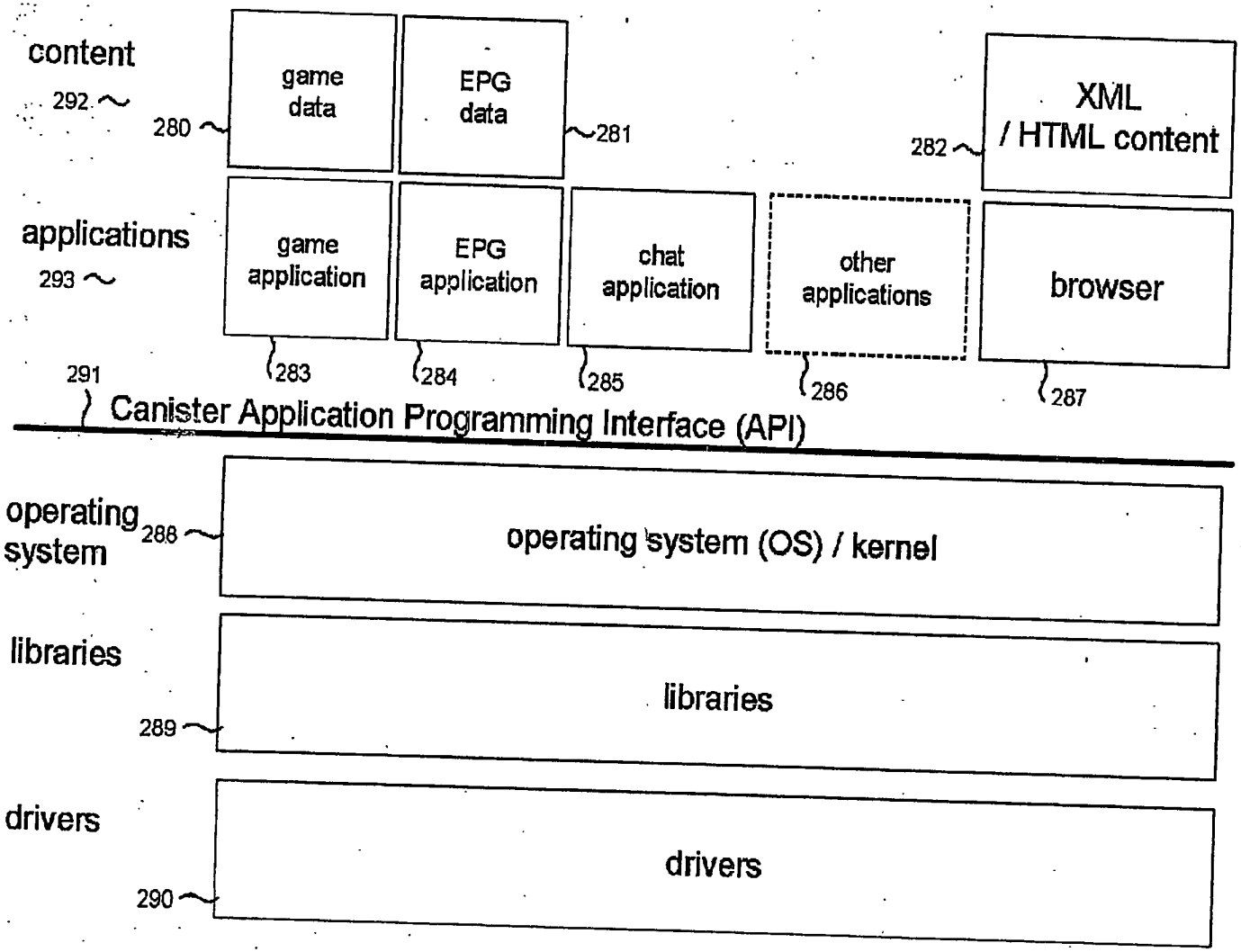

Figure 7: Structure of the memory canister

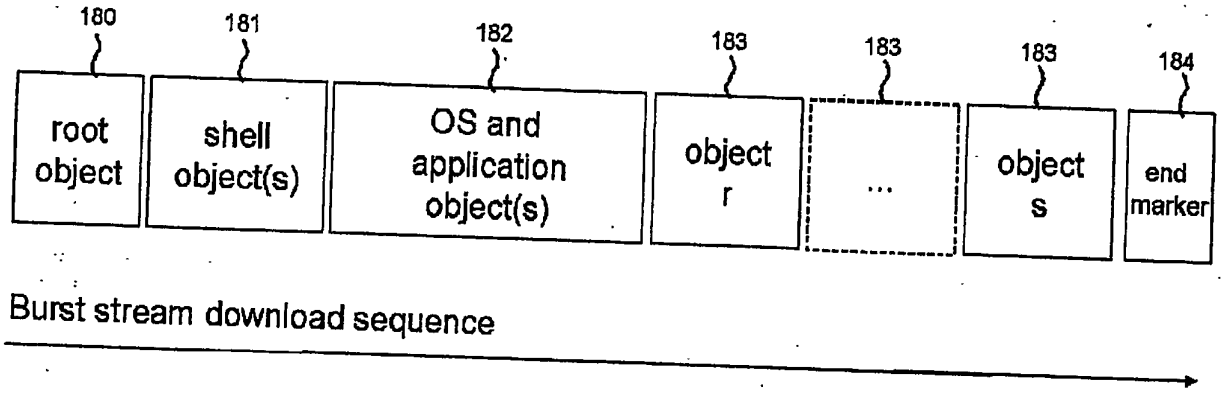

Figure 8: Burst stream download sequence

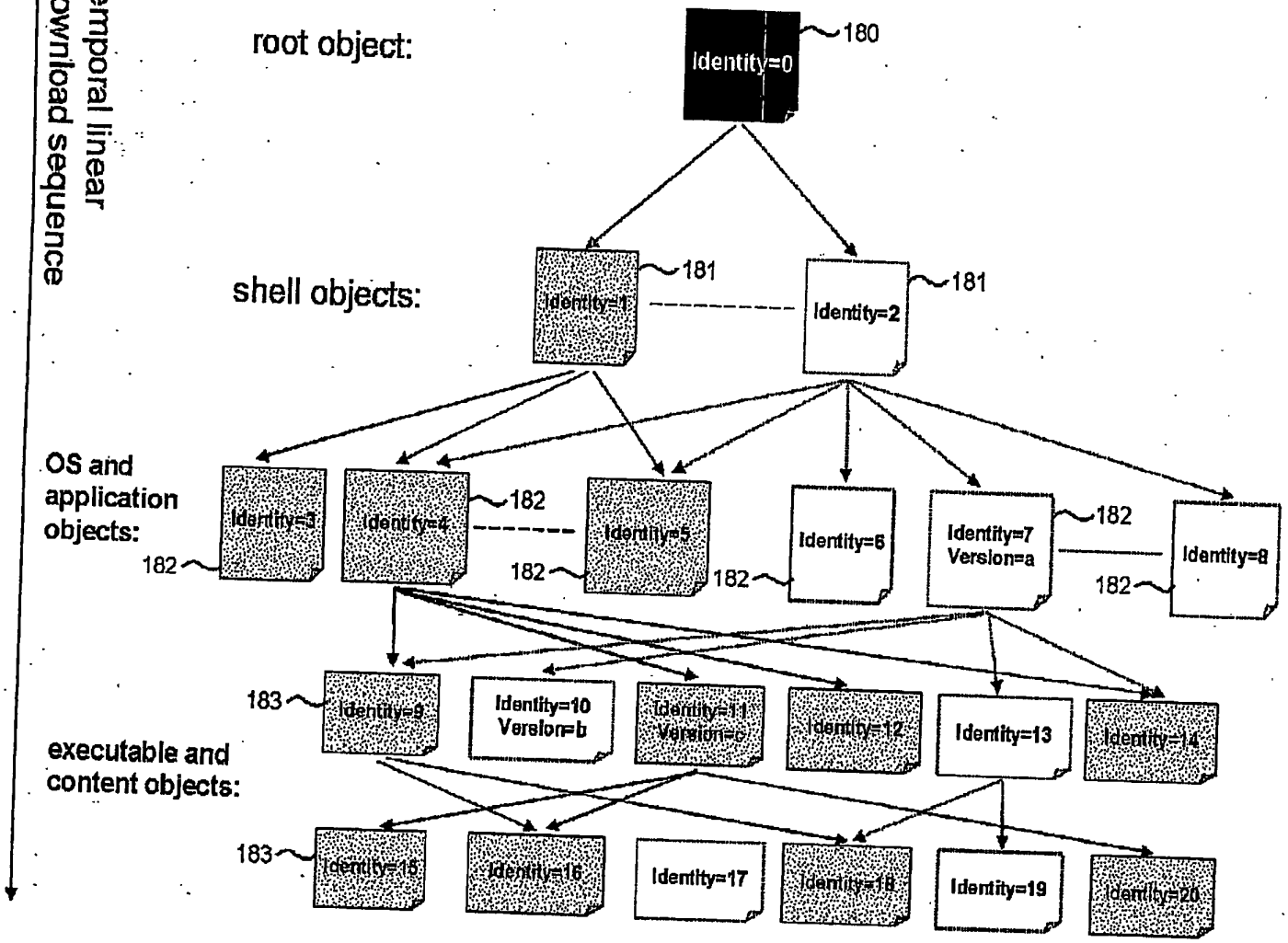

Figure 9: Burst stream data structure

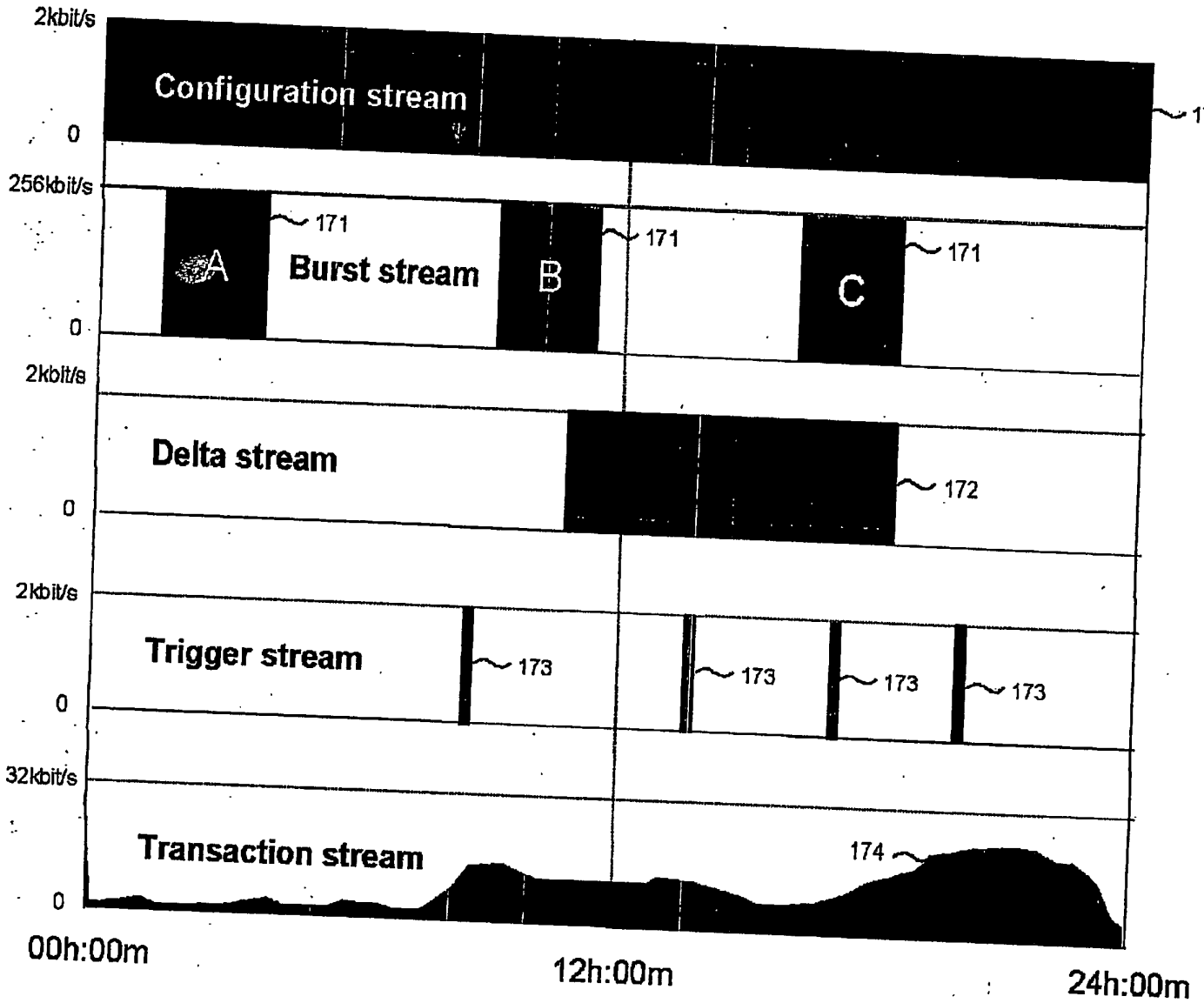

Figure 10: Typical data stream activity throughout a day

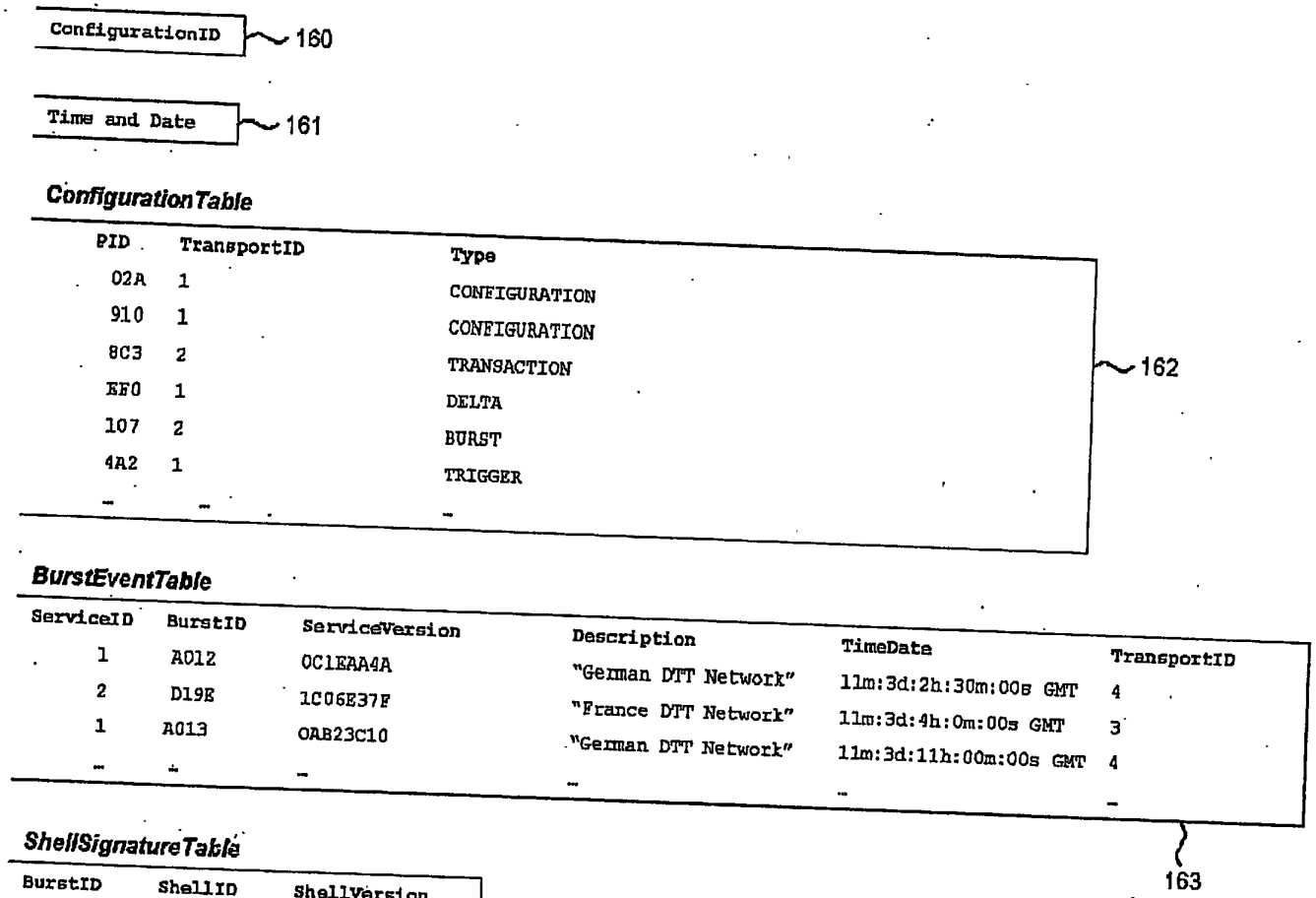

Figure 11: Configuration stream data structure

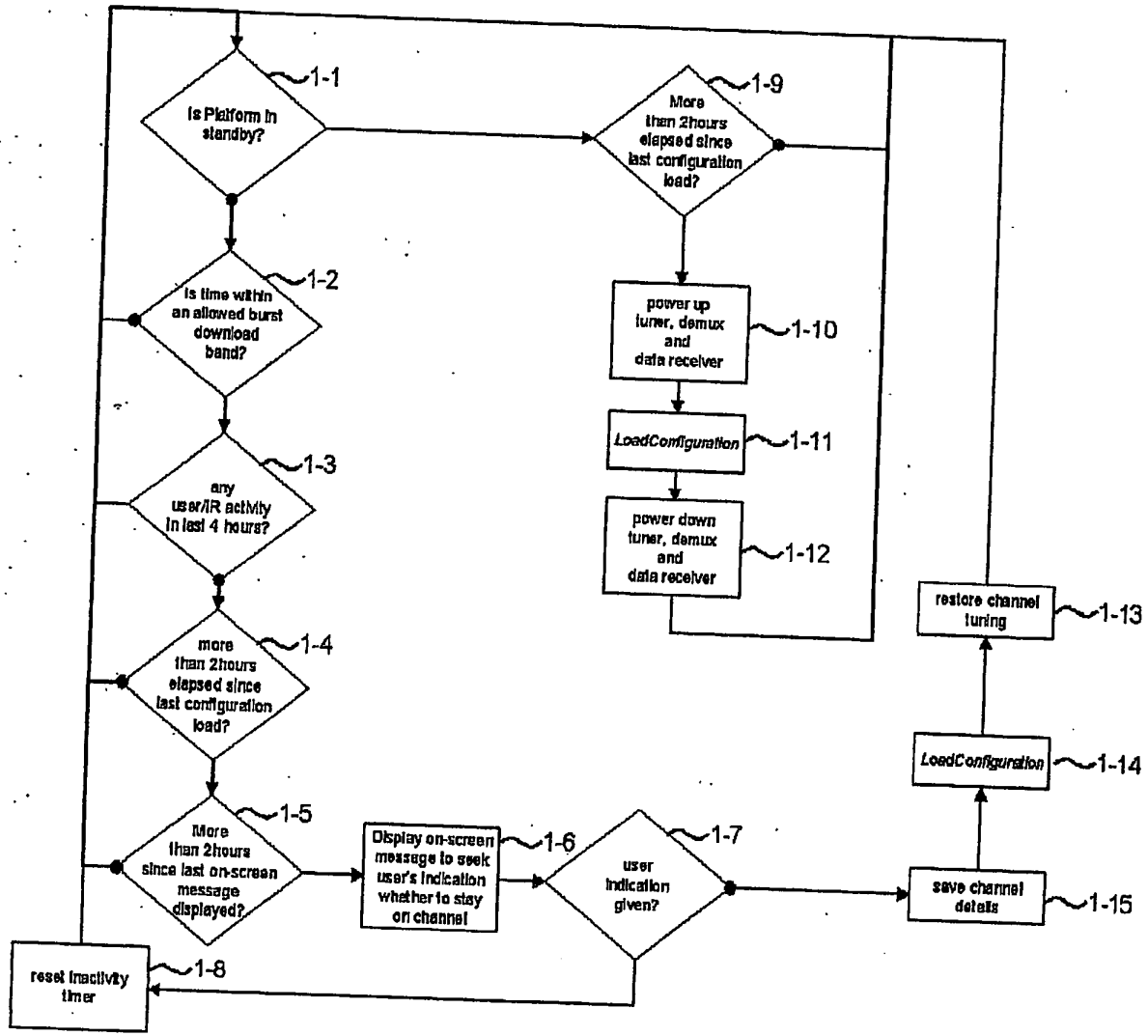

Figure 12: Configuration load loop

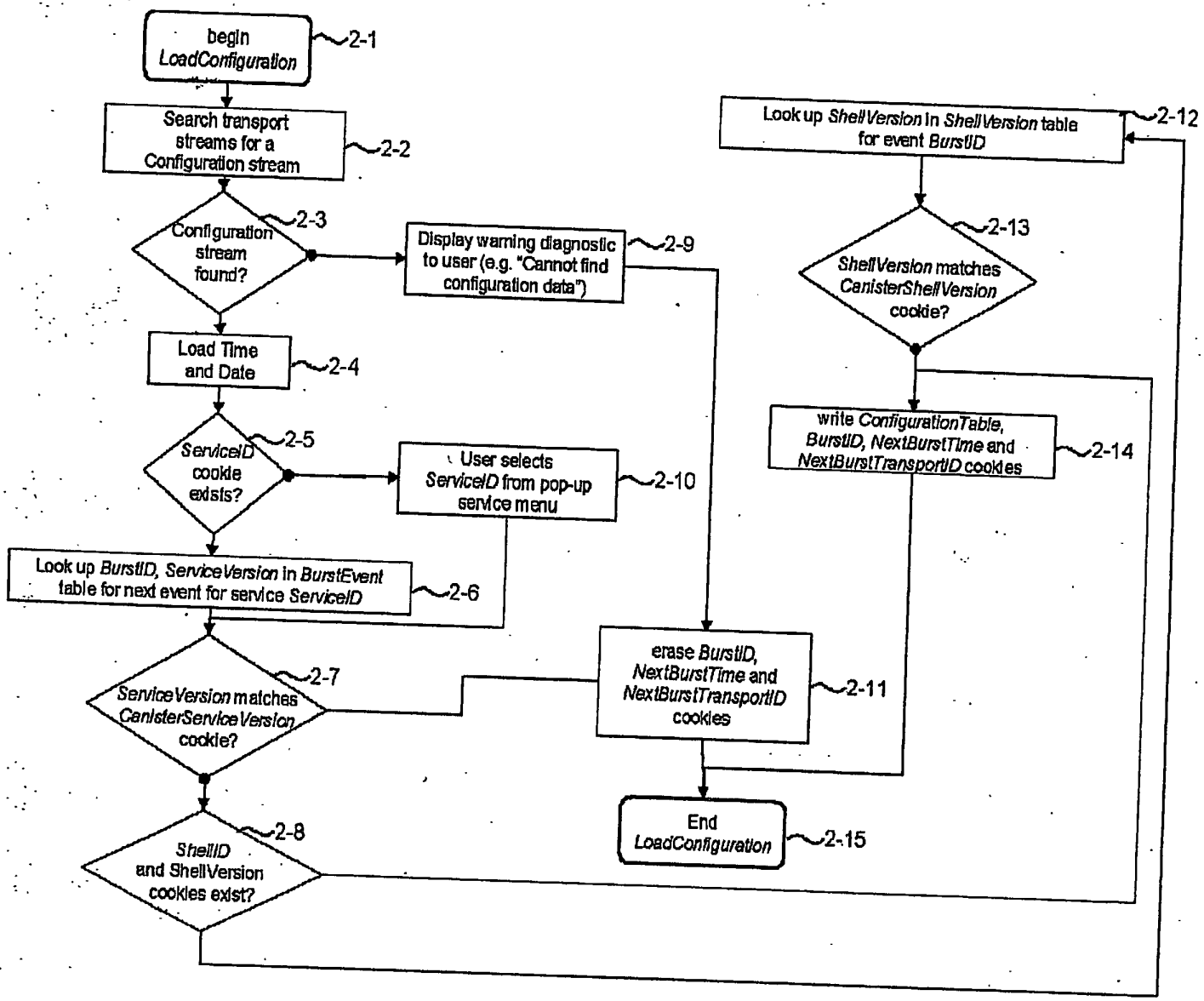

Figure 13: Determining parameters for next Burst event from Configuration stream

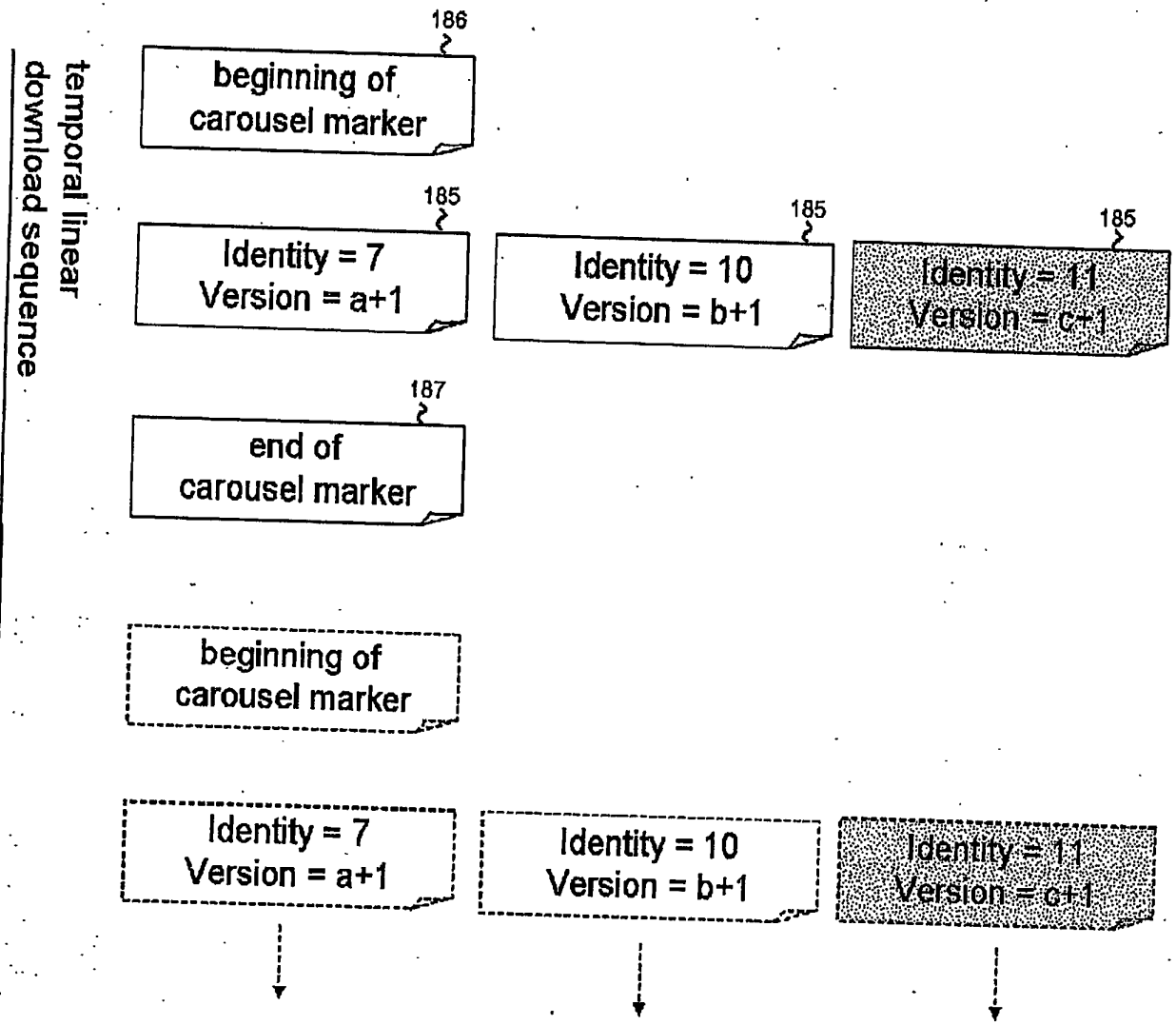

Figure 14: Delta stream data structure

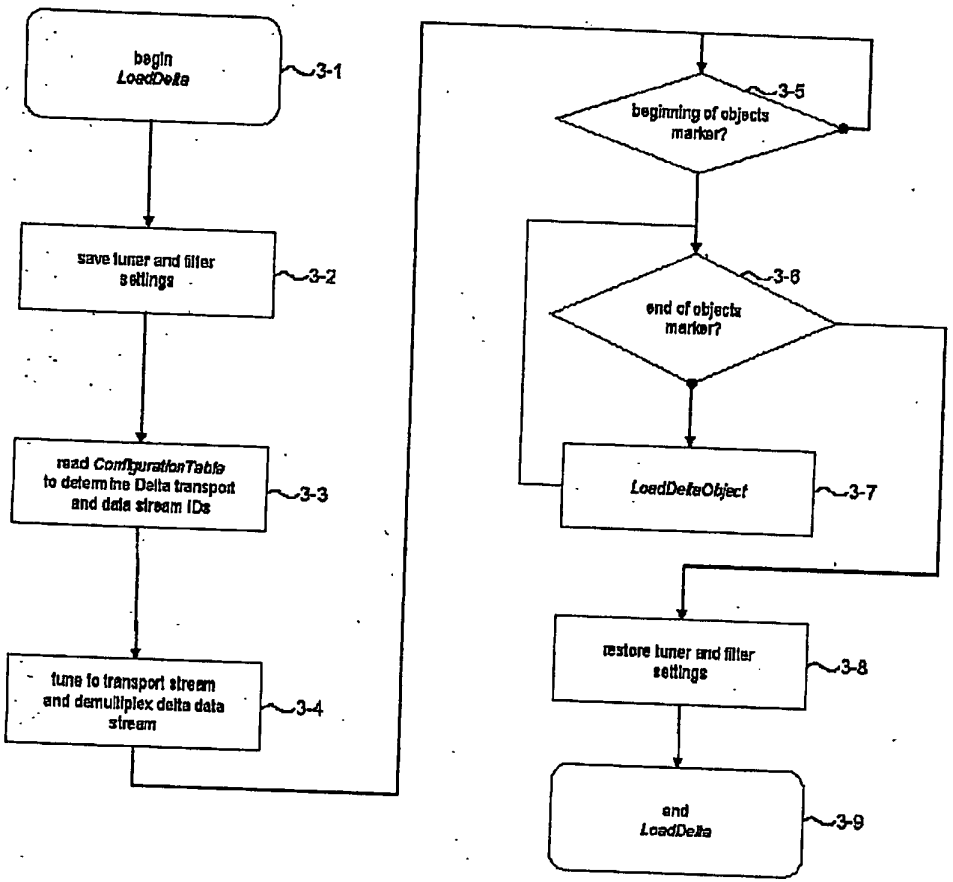

Figure 15: Object delta stream load loop.

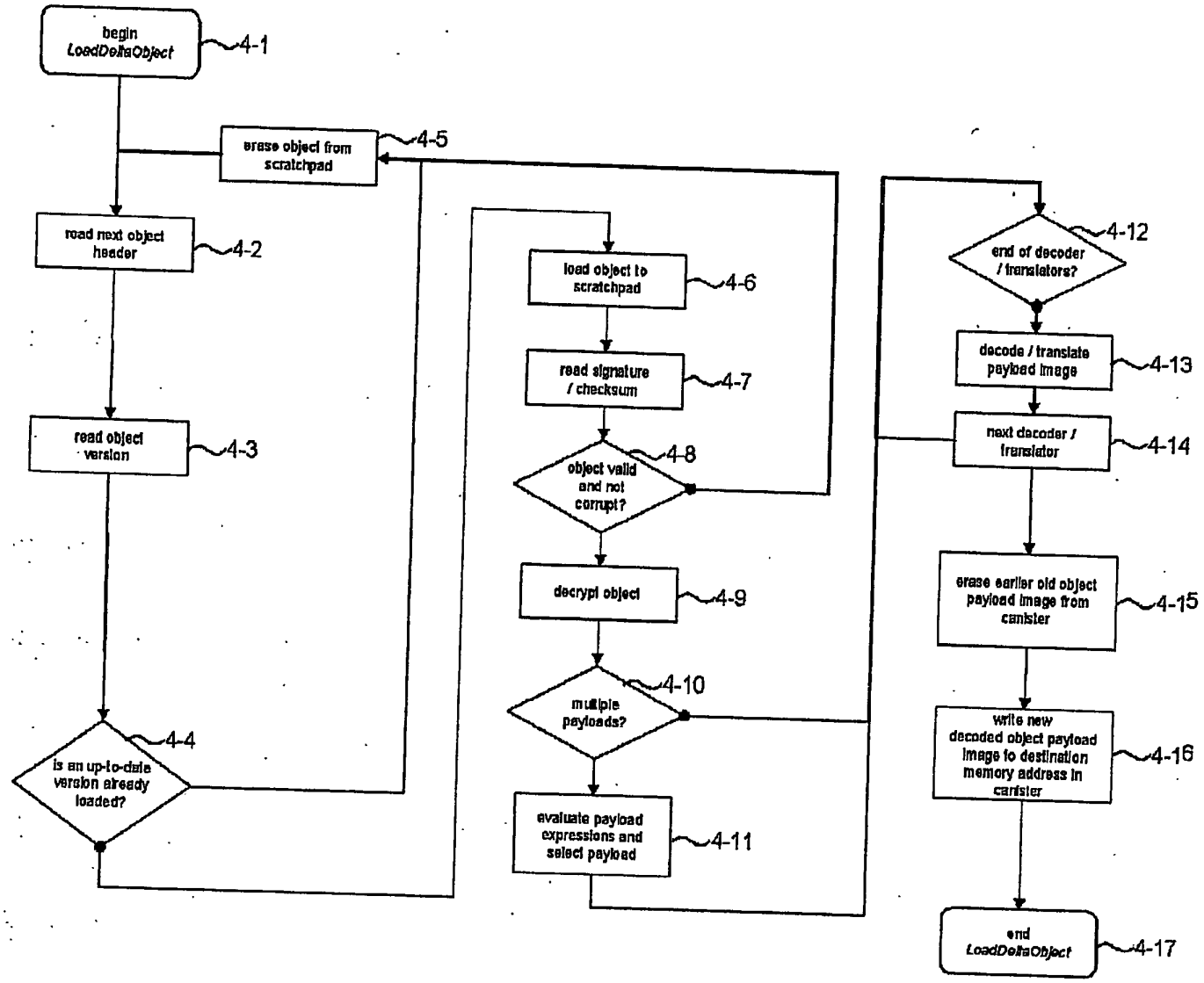

Figure 16: Object delta load process

6/13/2003 5:25 PM FROM: +441628602923 DRAZIN TO: 01633 814444 PAGE: 047 OF 067

Figure 17: Display of a conventional television service channel on screen.

0071745 13-Jun-03 05:54

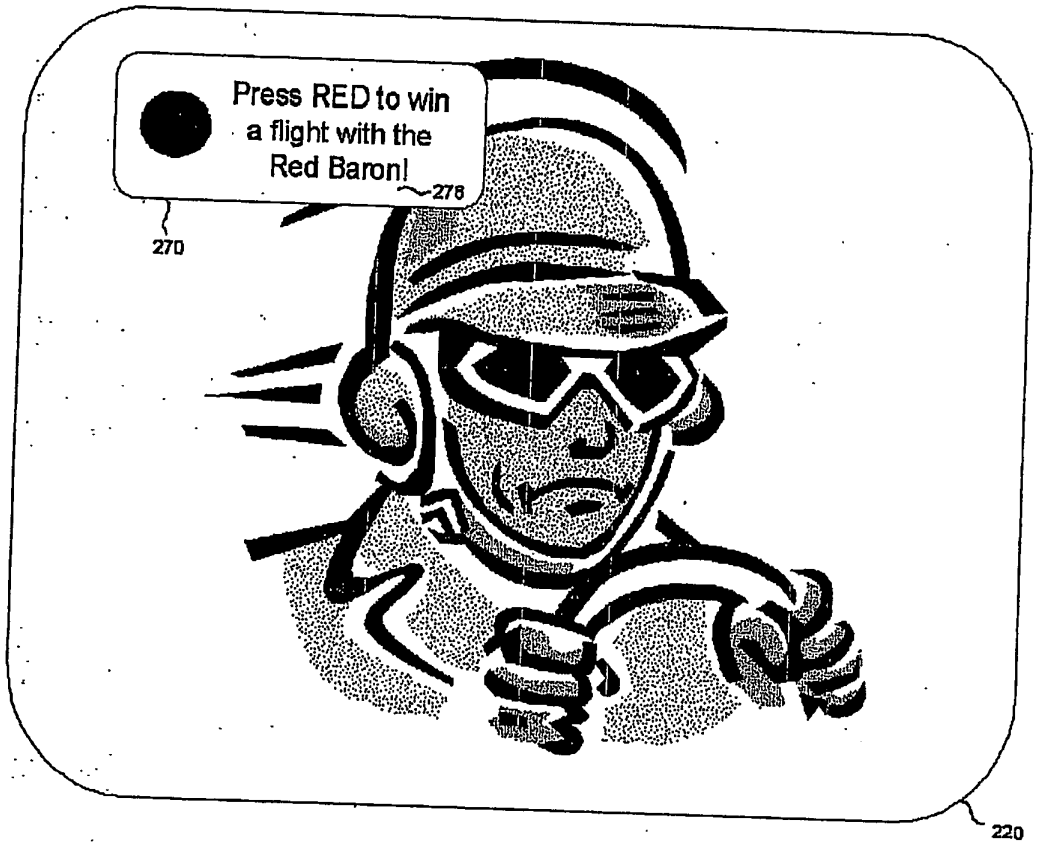

Figure 18: Display of a trigger prompt

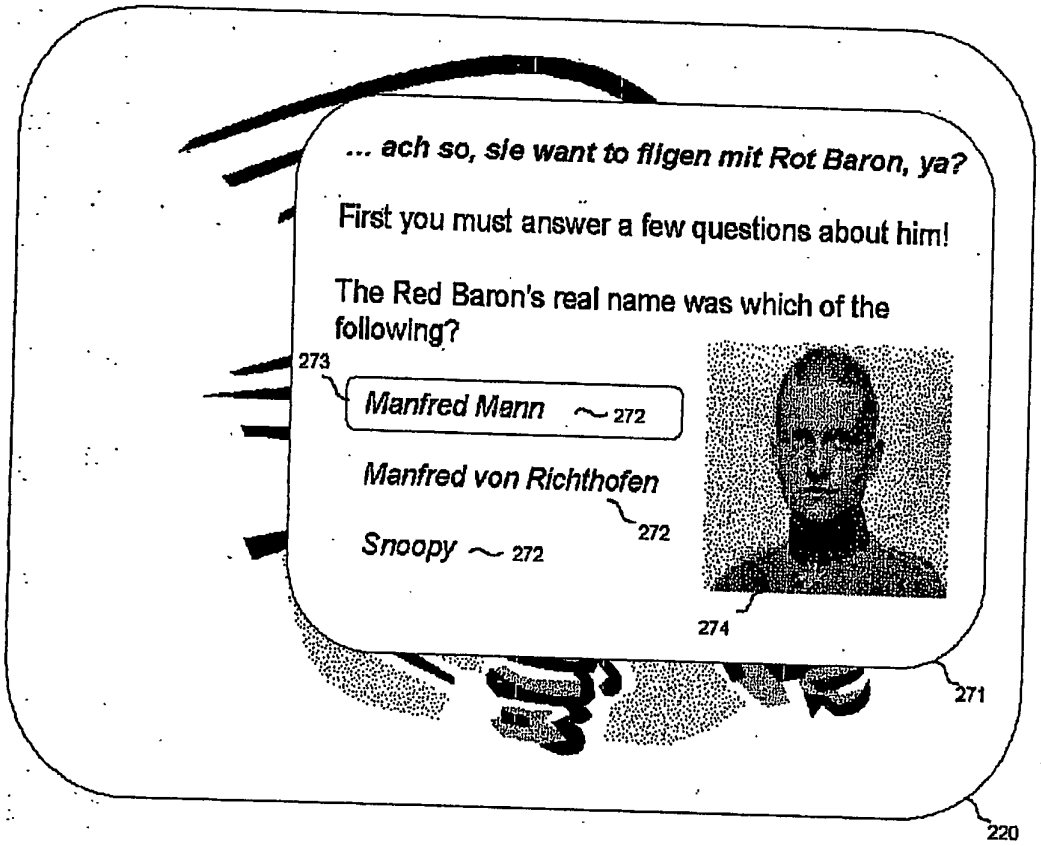

Figure 19: Interactive scenes triggered to enhance a TV programme or advertisment.

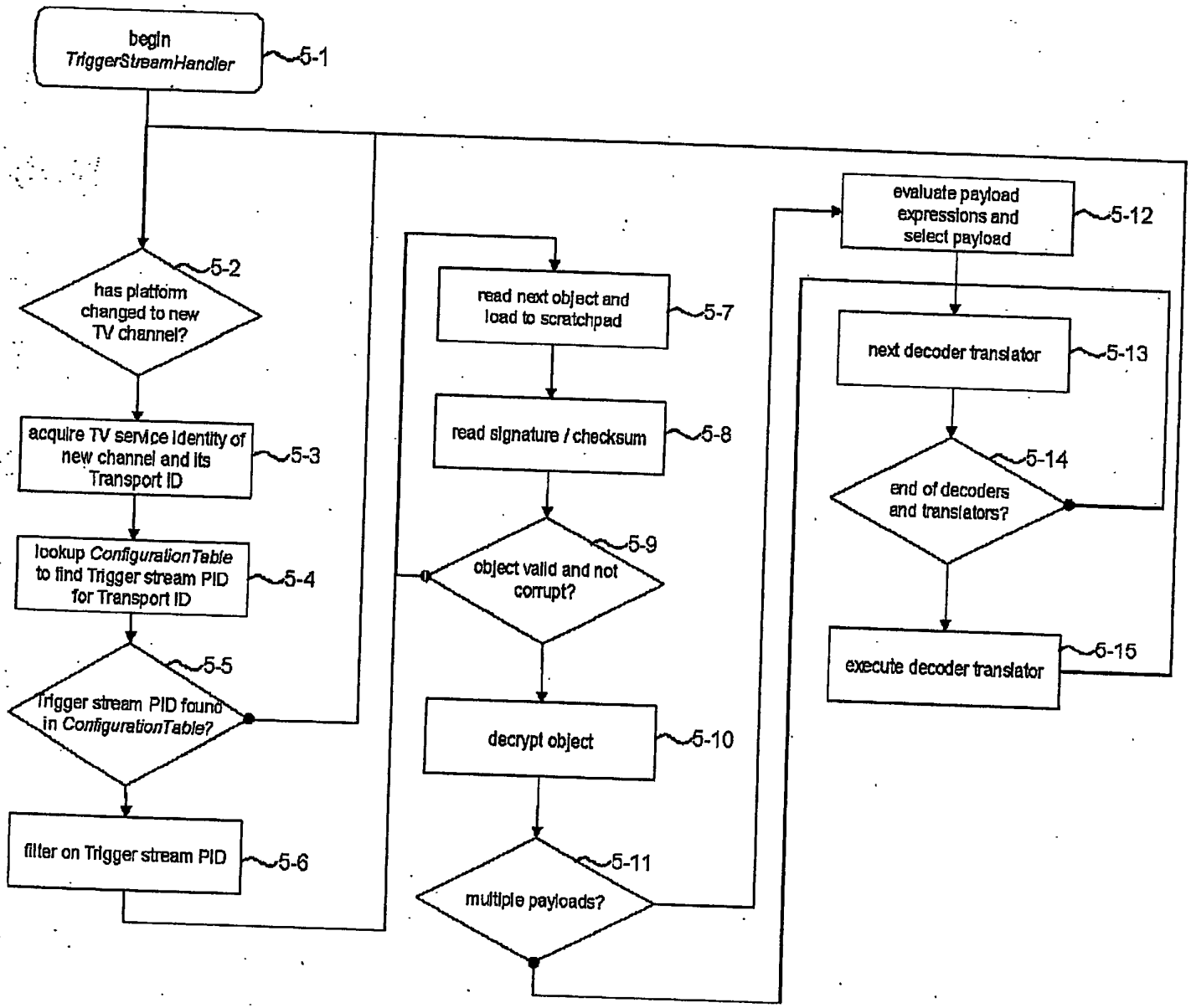

Figure 20: Trigger stream object handling process

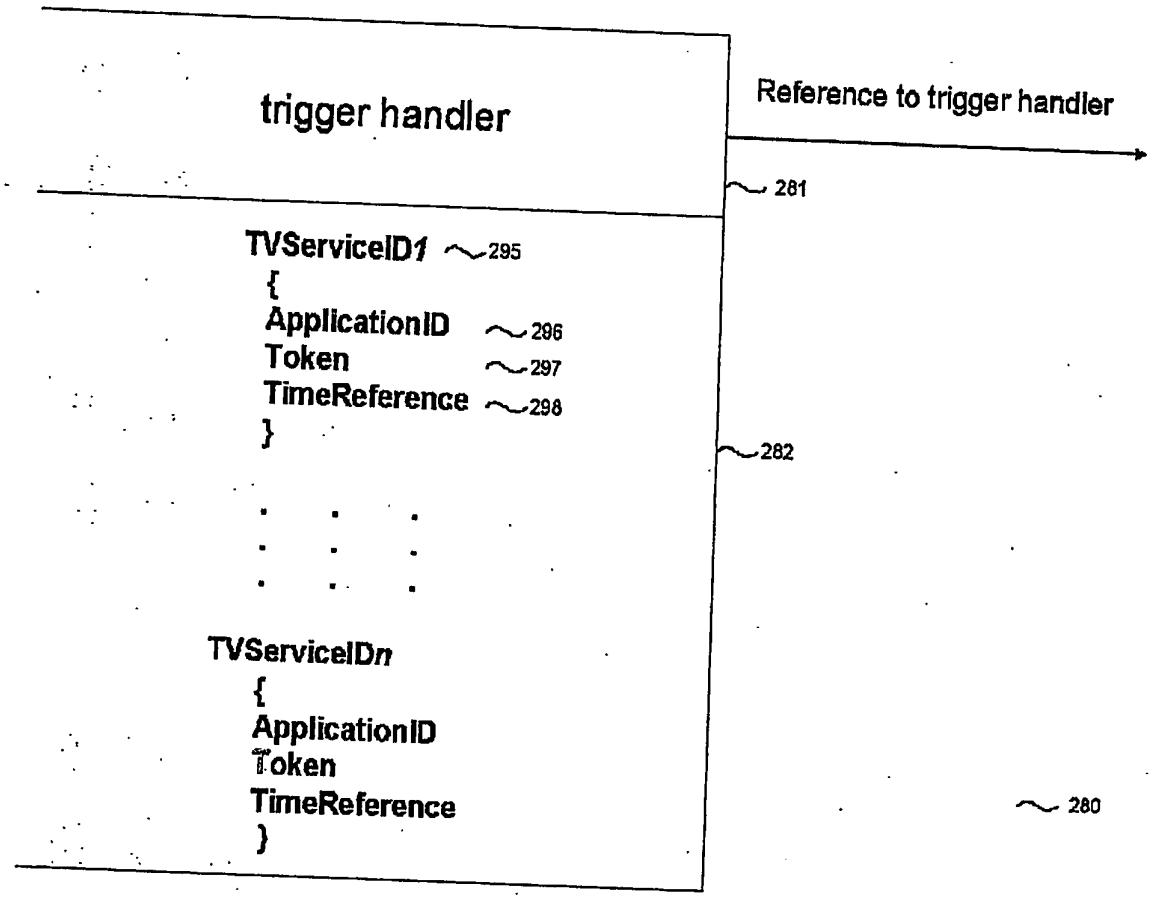

Figure 21: Trigger Payload Structure

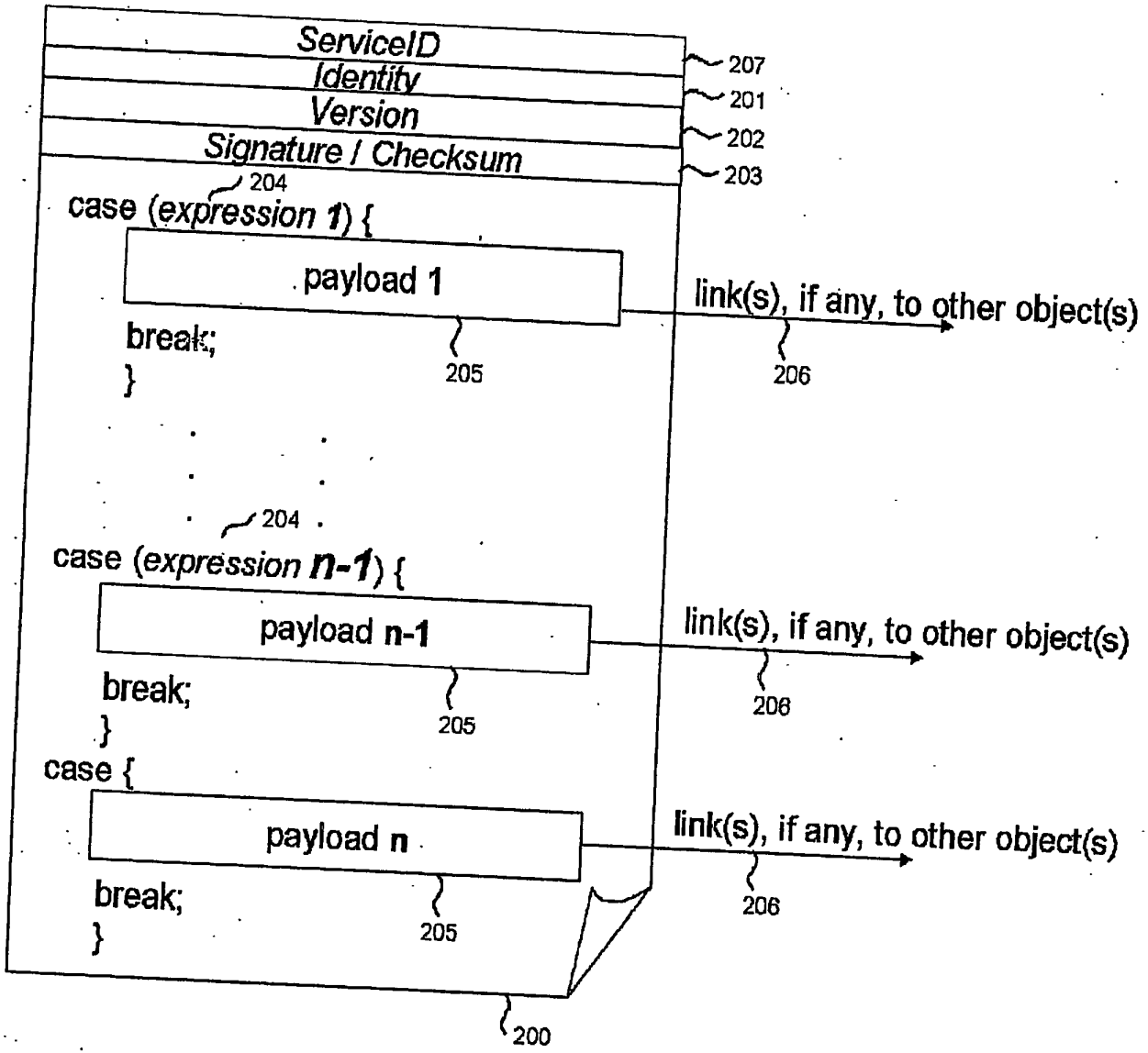

Figure 22: Object data structure

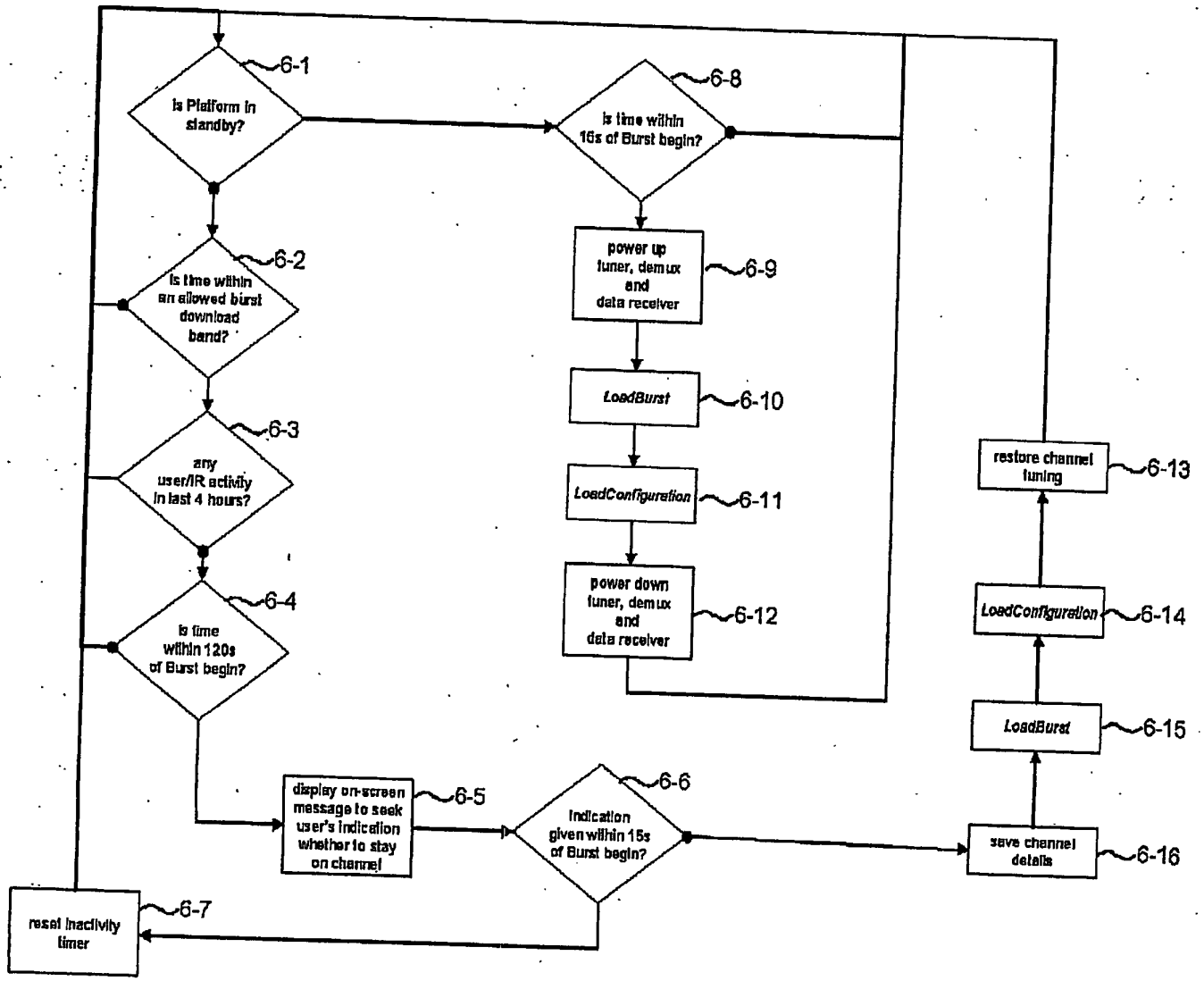

Figure 23: Burst load loop

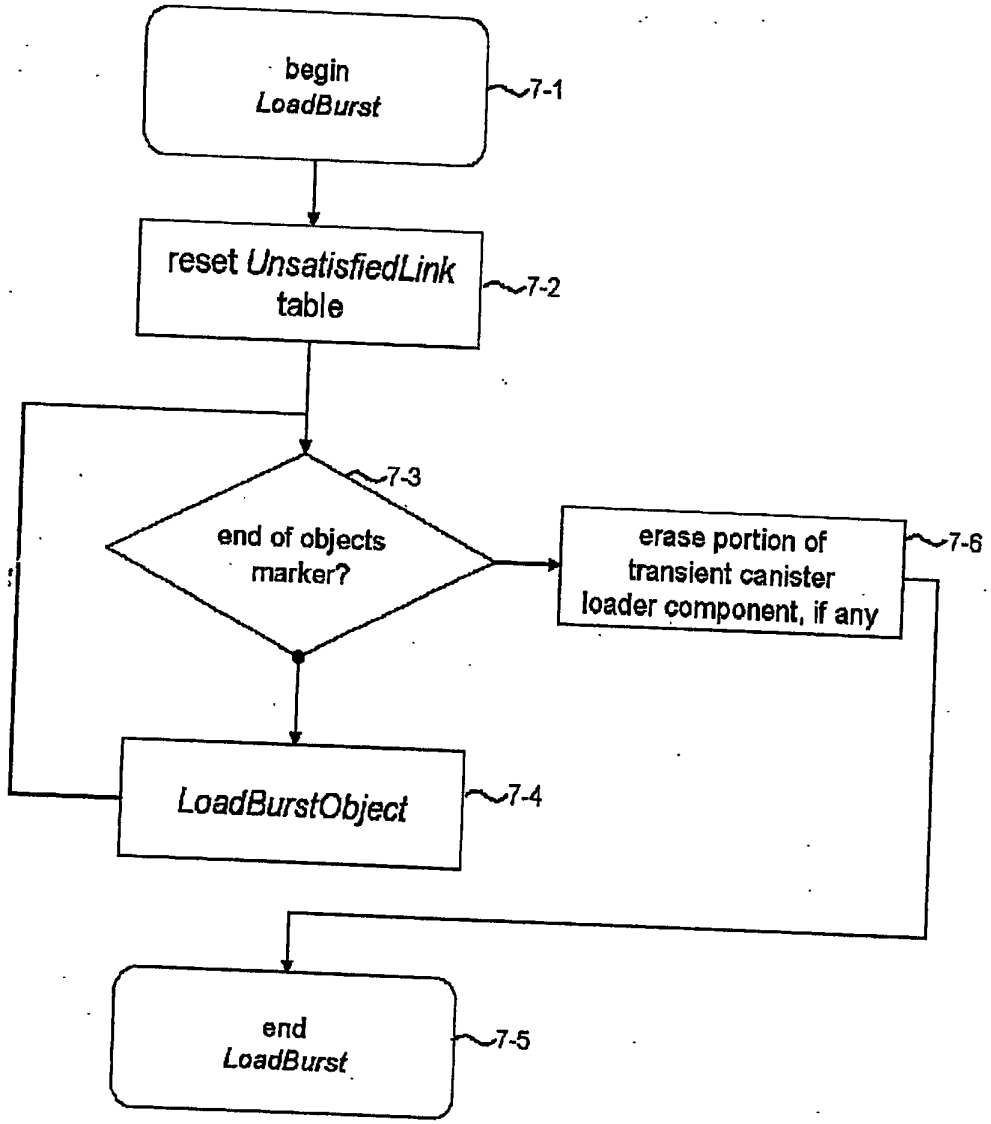

Figure 24: Object burst stream load loop

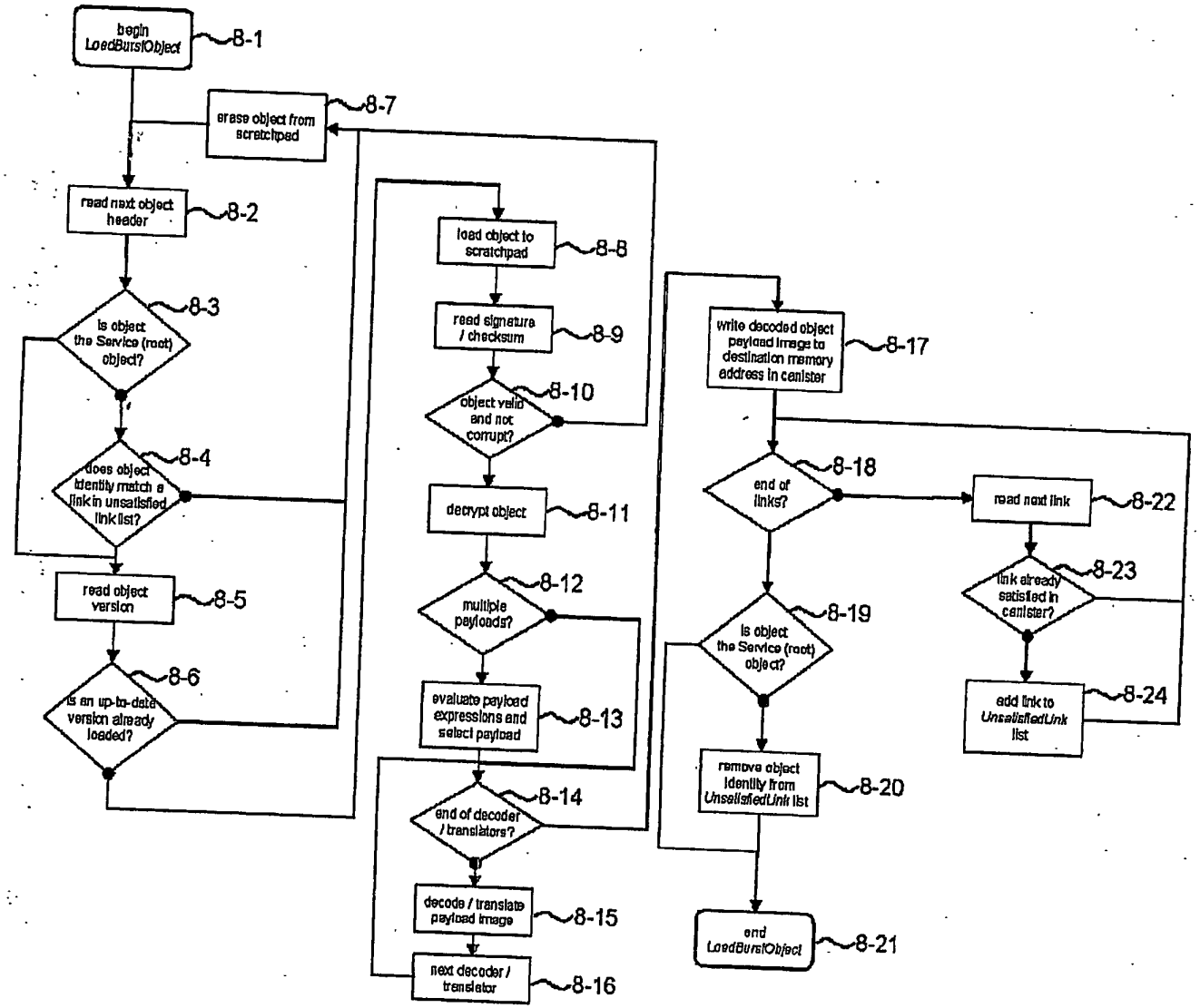

Figure 25: Object burst load process

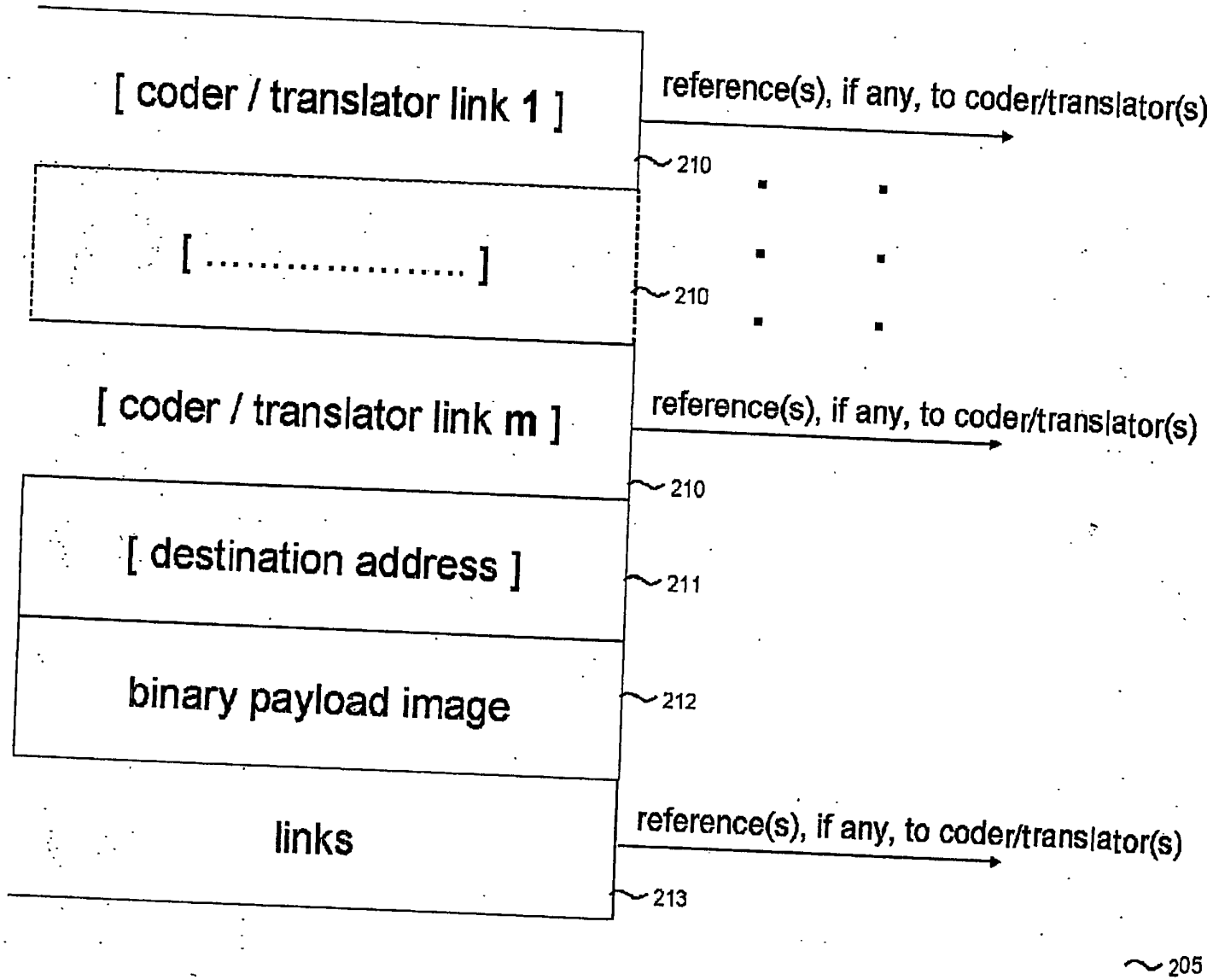

Figure 26: Structure of object payloads

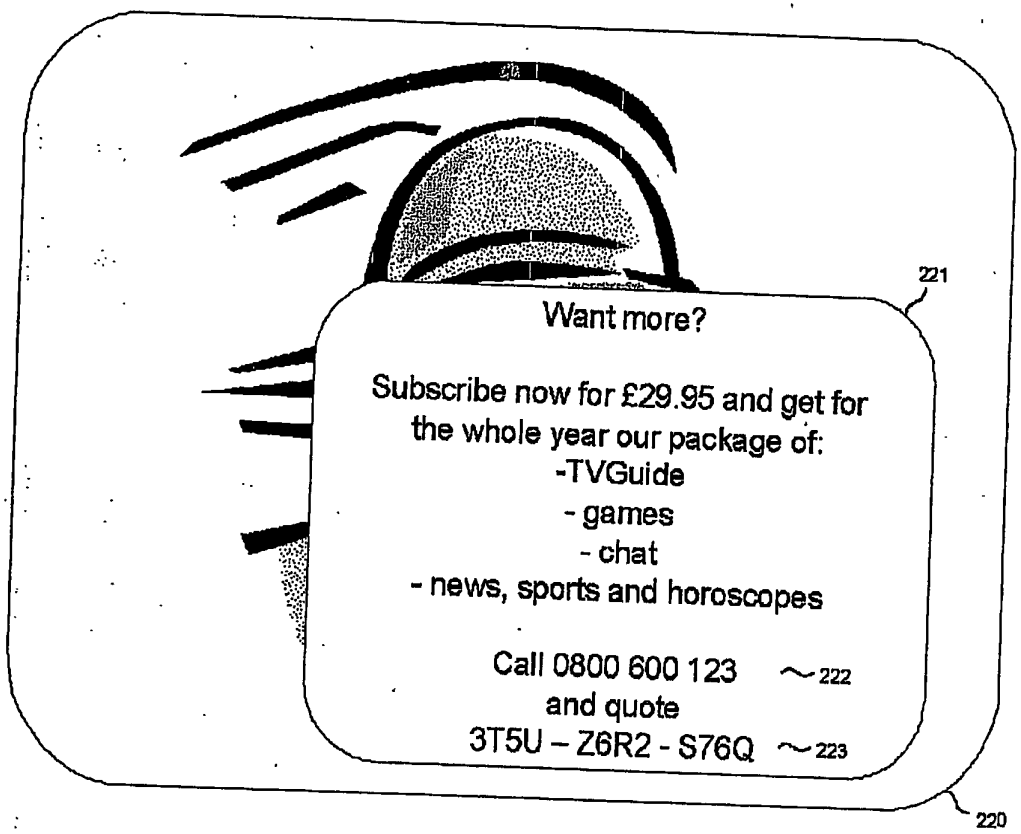

Figure 27: Overlay of invitation to subscribe over television picture

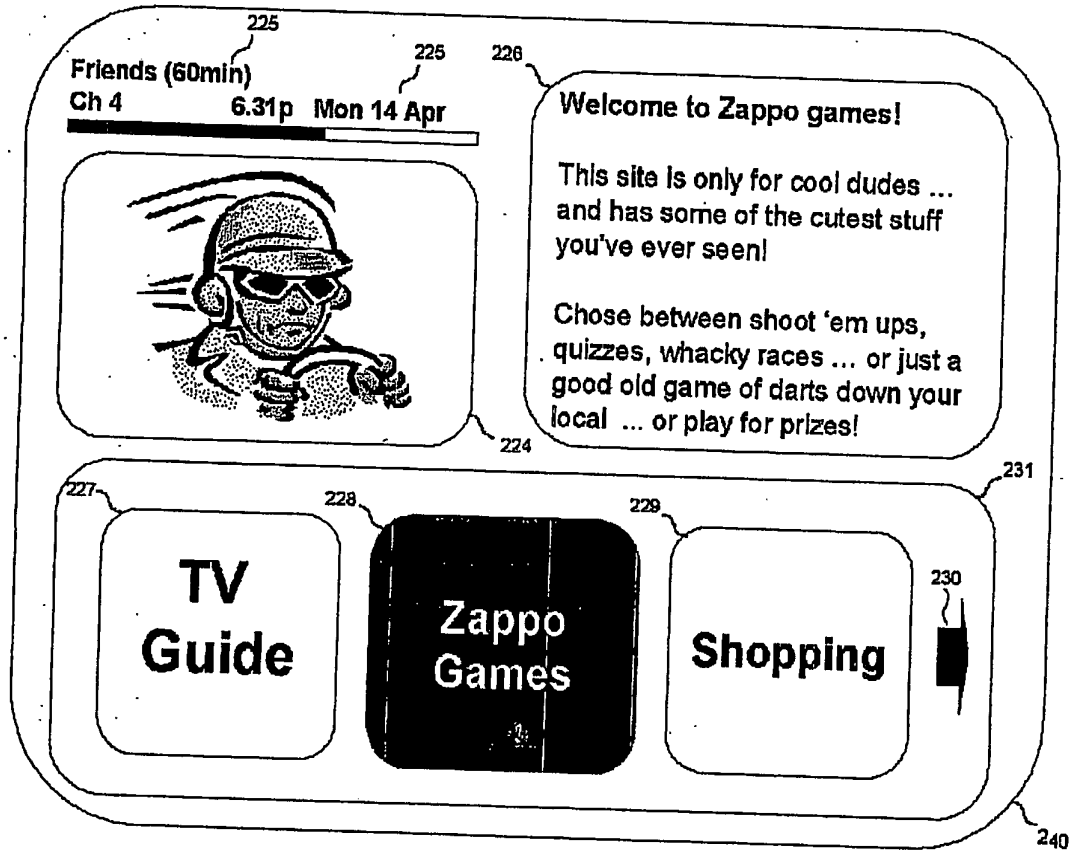

Figure 28: Browsing between canister services

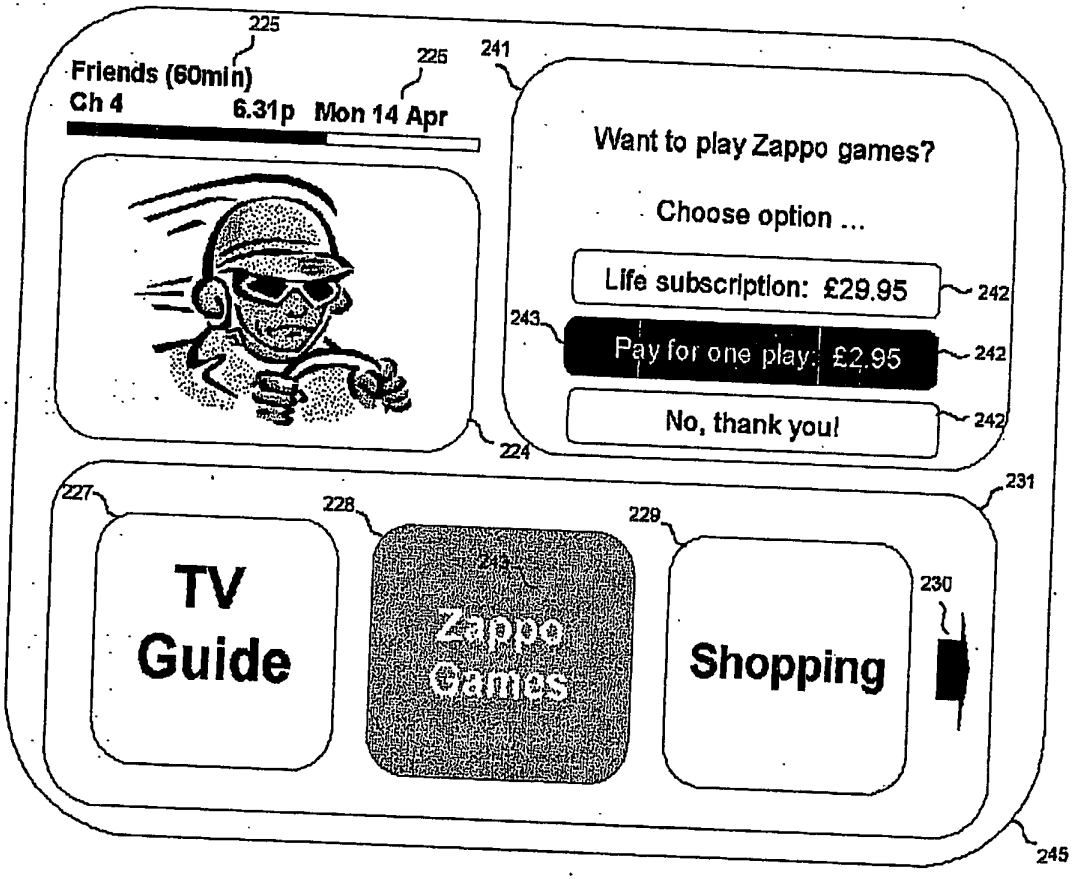

Figure 29: Choosing an entitlement option to access a canister application

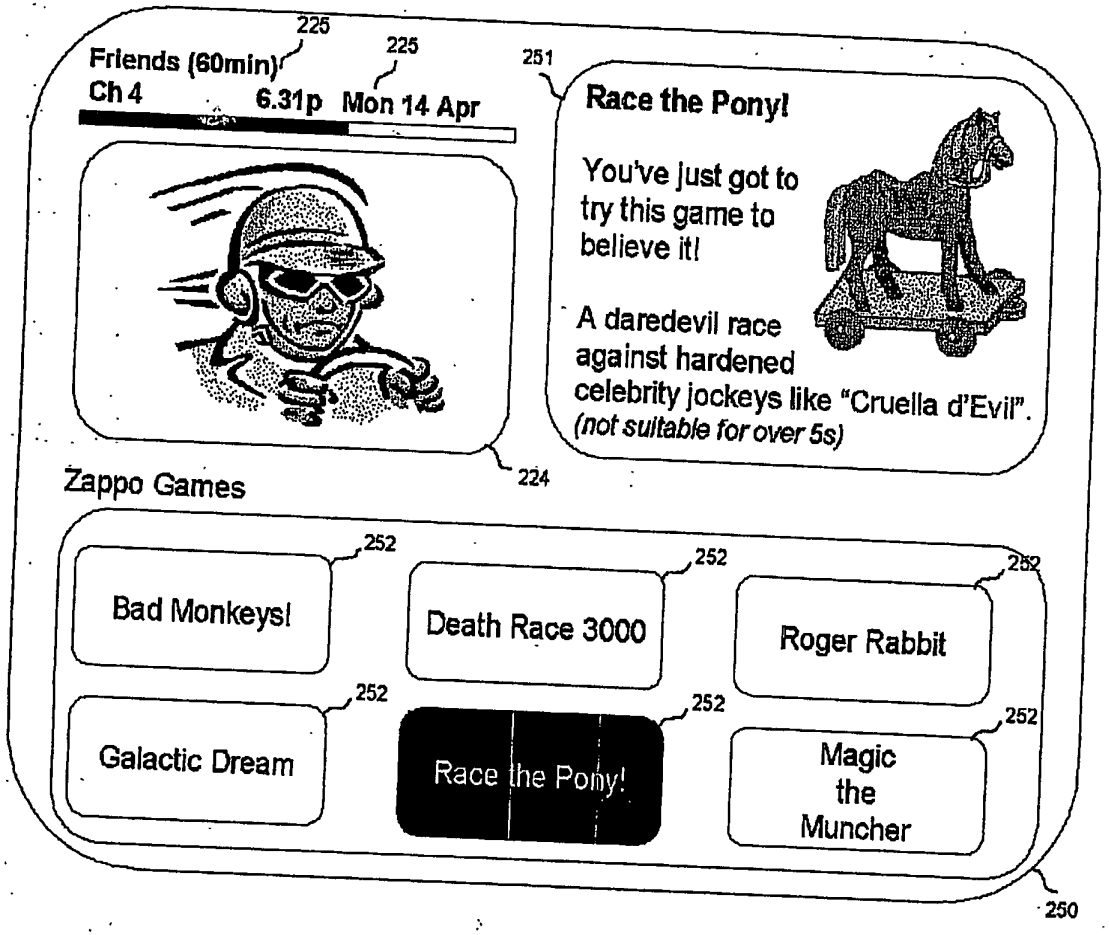

Figure 30: Selection of a content title or service feature

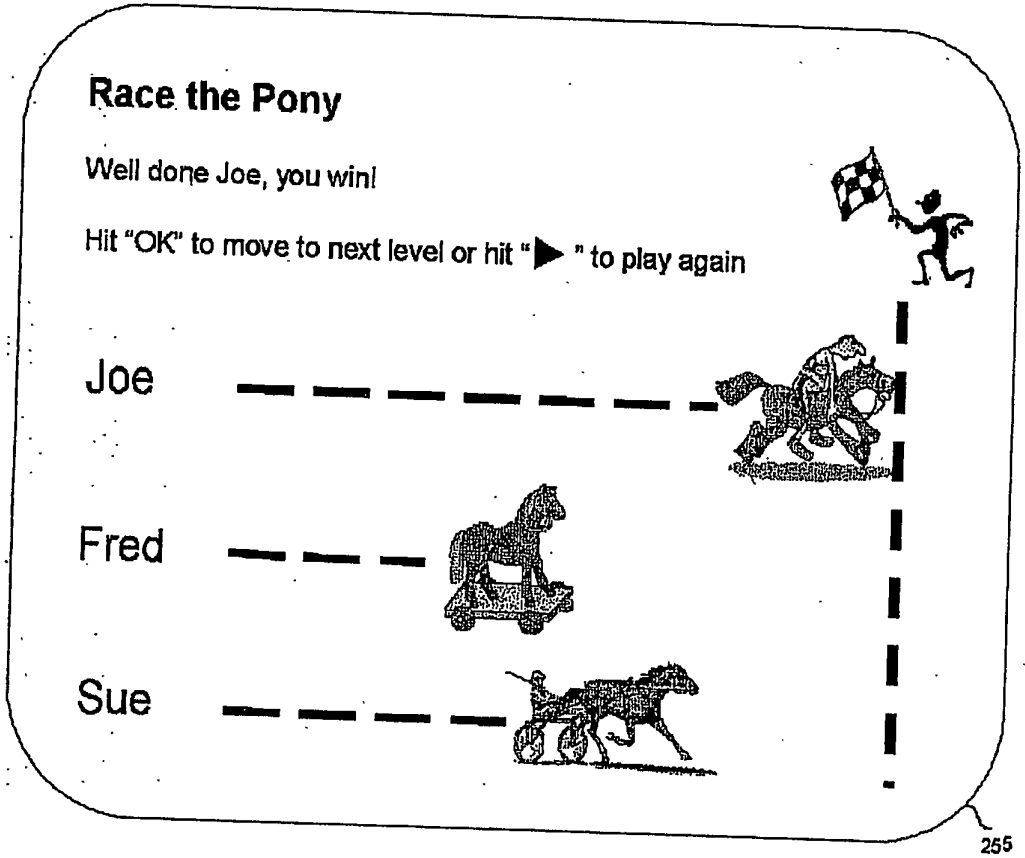

Figure 31: Game application where user meets a criterion to be entitled to access new functionality

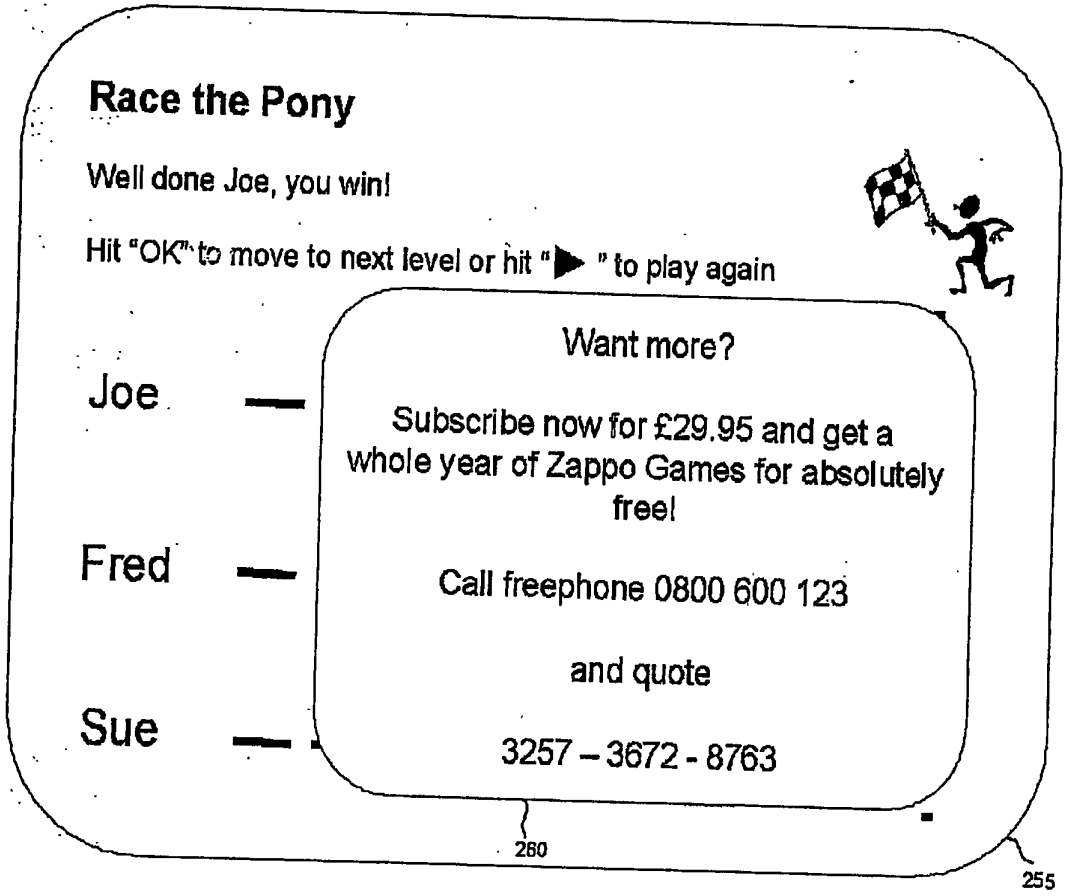

Figure 32: Overlay of an invitation to pay or subscribe over a service screen

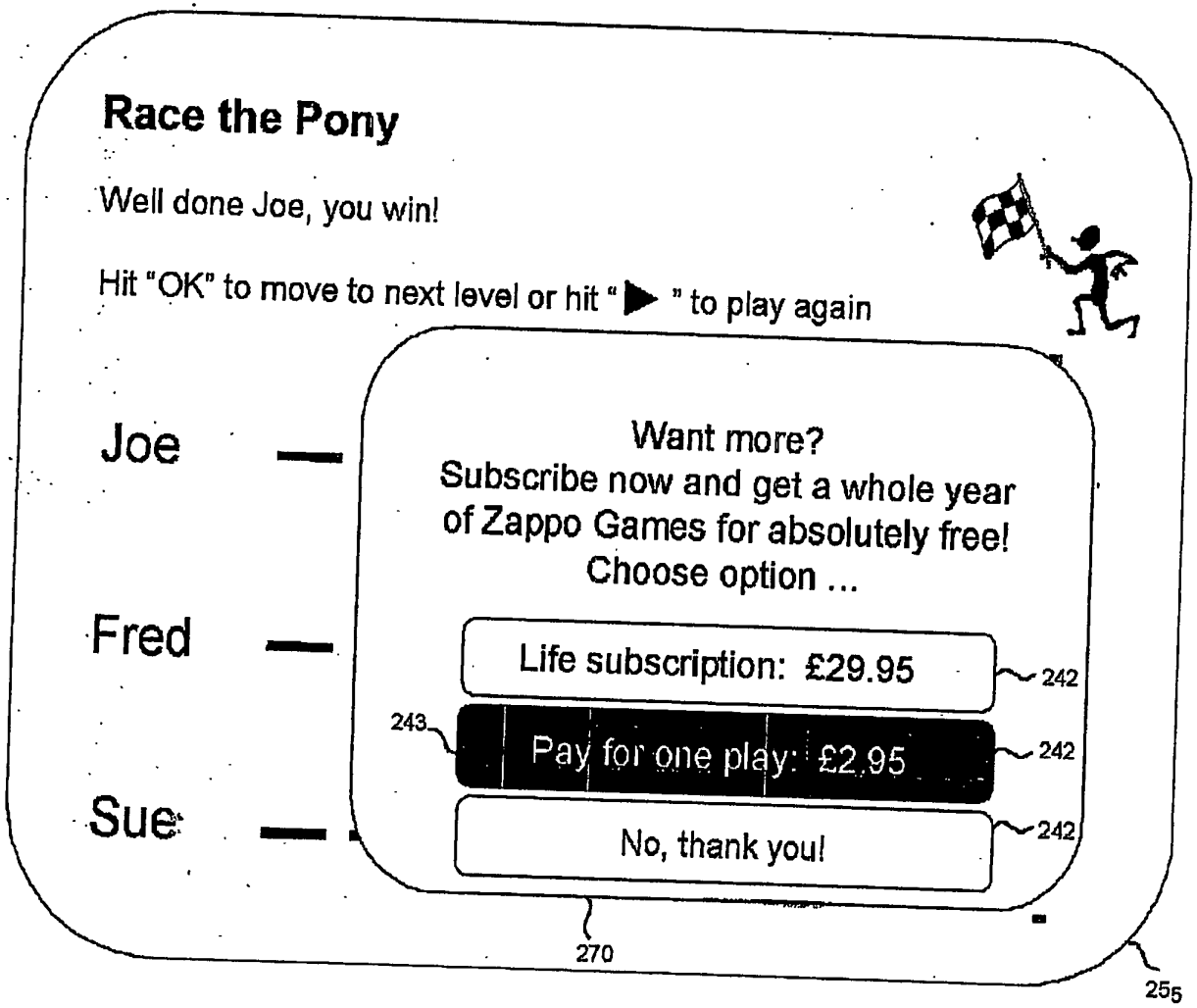

Figure 33: Overlay of an invitation to pay or subscribe over a service screen that contains user selectable components

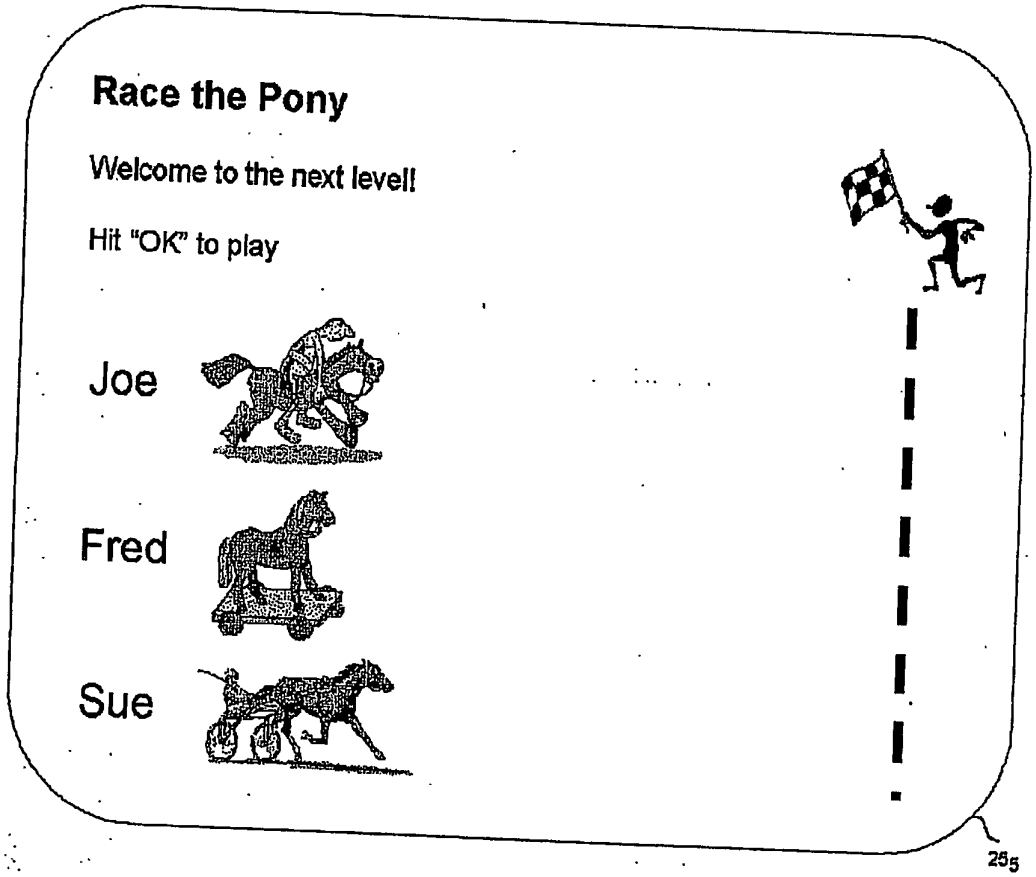

Figure 34: Screen describing the enablement of a feature or content option

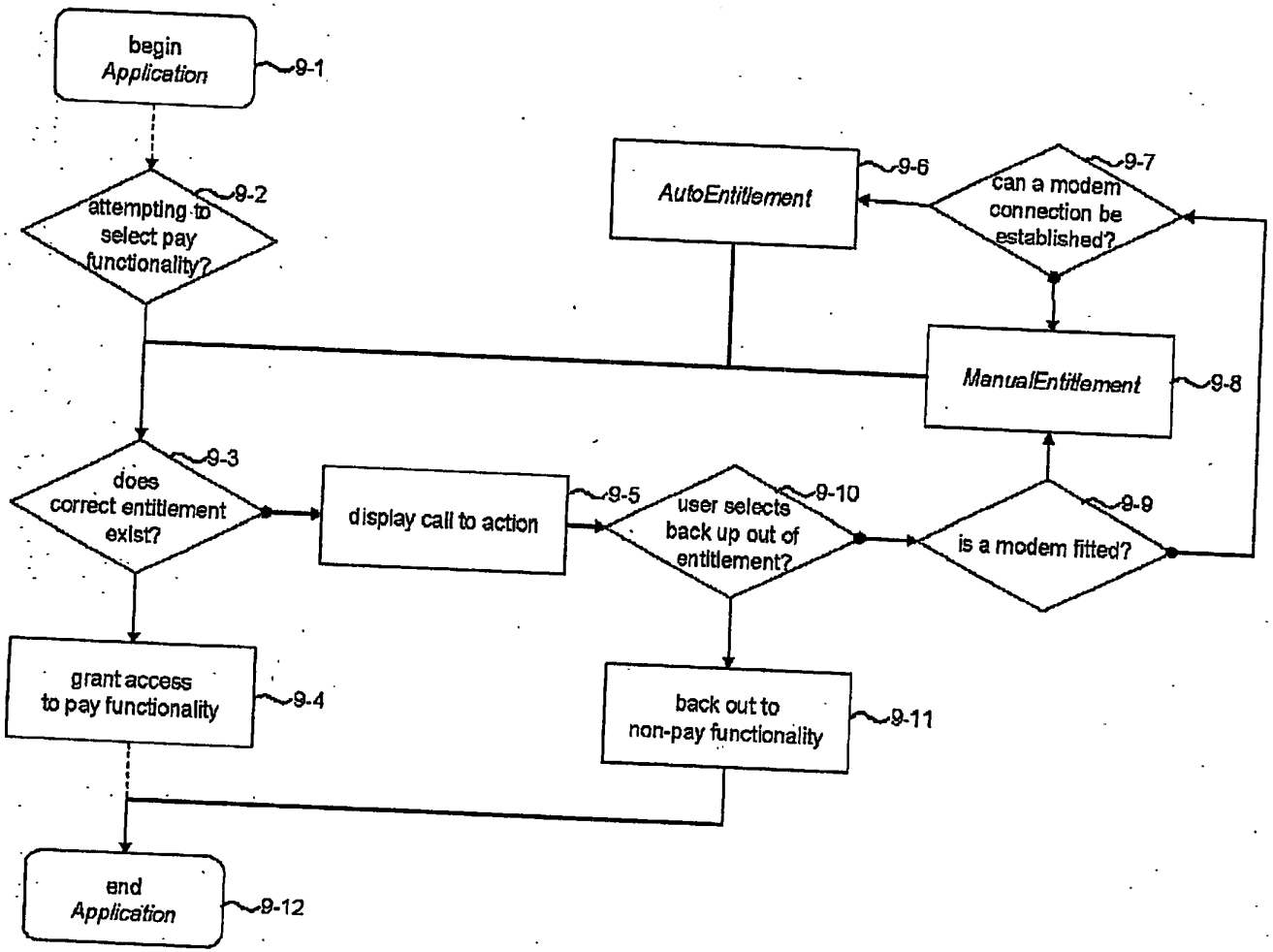

Figure 35: Process whereby applications invite a call to action

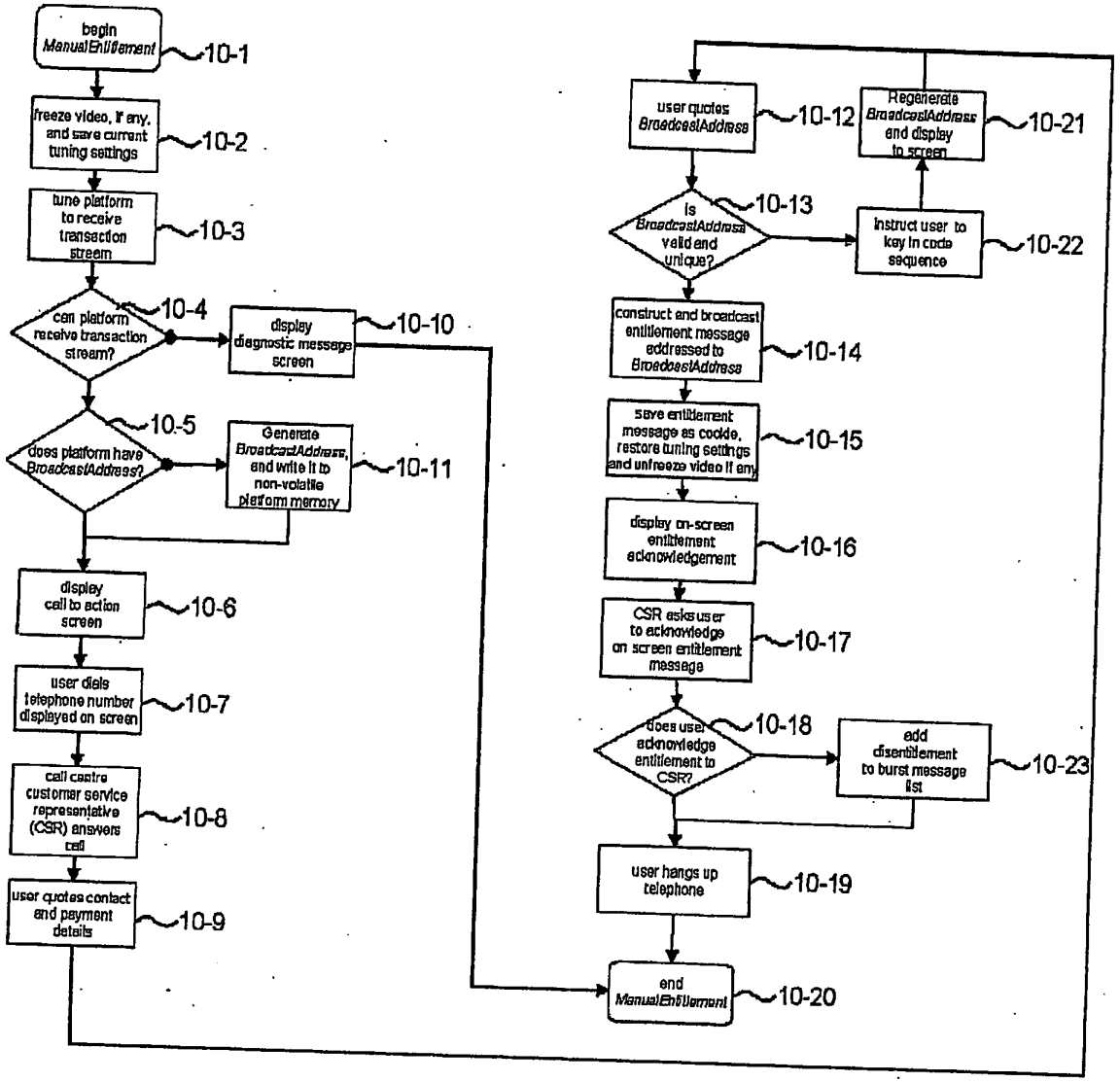

Figure 36: Process for manually entitling a platform to receive a service

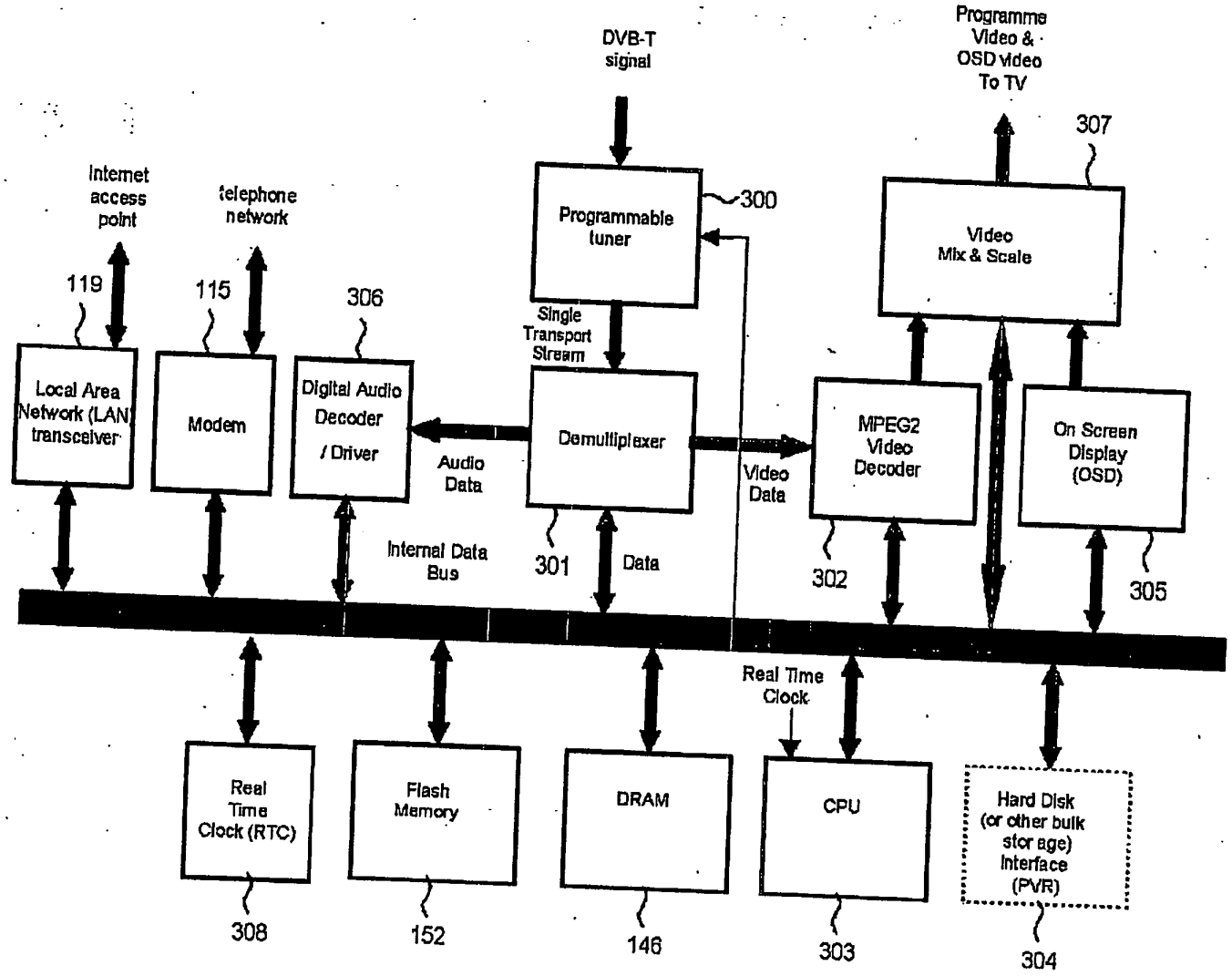

Figure 37: Set-top-box hardware